Ingeniería Técnica en Informática de Gestión

PROYECTO FIN DE CARRERA

Sincronizador de tareas entre aplicación de escritorio y servicio web

Autor: Alejandro López Ruiz
Tutor: David Palomar Delgado

Colmenarejo, Febrero de 2013

Título: Sincronizador de tareas entre aplicación de escritorio y servicio web
Autor: Alejandro López Ruiz
Director: David Palomar Delgado

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 16 de Febrero de 2013 en Colmenarejo, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

Agradezco a mi familia que me ha apoyado desde el principio de la carrera hasta el final con las mismas fuerzas, siempre.

Agradezco a todos mis amigos, y en especial a mis “hermanos” por darme ánimo y sacarme del agobio con una facilidad increíble.

Y por supuesto a los numerosos muses jugados con los compañeros de clase, en el tiempo en el que deberíamos de estar en clase.

“La culpabilidad es un invento humano”

Alejandro Jodorowsky

Resumen

Internet, desde su creación, ha evolucionado a pasos de gigante y ha permitido que la sociedad en la que vivimos pueda disponer de más información que en cualquier otra época. Esto ha permitido, en gran medida, que podamos vivir mucho mejor y que las nuevas tecnologías se adapten a este gigante, porque es lo que realmente la gente utiliza día a día, ya sea en su trabajo o en su tiempo libre.

En un principio todo era muy simple y con muy pocas posibilidades, páginas en las que no podías escribir nada, solamente el creador las podía modificar y escribir a su antojo, hasta que evolucionó hasta un punto en el que podíamos hasta ver vídeos que podías incorporar en la página web que tú mismo habías creado. Ya no solo eso, si no disponías de conocimientos para poder realizar tu página web, algunas empresas ponían a tu disposición una nueva forma de crear tu espacio en internet donde poder organizar este espacio a tu gusto: los Blogs. Y no se quedo ahí la cosa, si no que además se creó una sociedad en internet en la que puedes tener a tus amigos al alcance de un click y mantener una conversación con ellos en tiempo real, y subir tus fotos y vídeos para compartirlos con ellos y plasmen su opinión. Y tampoco se quedo ahí, si no que a partir de ahora puedes conectarte a esa gran fuente de información a través de tu móvil y disponer de ella en cualquier lugar y a cualquier hora.

Todo esto de lo que acabamos de hablar está relacionado con el aspecto social, pero ¿Cómo ha contribuido internet a enriquecer el mundo empresarial y laboral? De una manera increíble. Gracias a internet es posible enviar documentos relacionados con la empresa como gráficos, estadísticas, inventarios, planes de empresa y muchos otros documentos a otros departamentos que pueden estar distribuidos por el mundo entero, y conectarlos de una manera increíble y en muy poco tiempo, cosa que antes se creía impensable. Podemos buscar trabajo gracias a internet y enviar nuestro currículum a una enorme cantidad de empresas sin tener que desplazarnos de casa. Internet ha ayudado a empresas y a trabajadores desde los principios de su creación.

Y el último invento revolucionario al que cualquier empresa puede sacar un provecho enorme son los gadgets y widgets que aumentan la comodidad ya que, a estas alturas ni siquiera necesitas visitar la página web, sino que dispones de un pequeño programa que está en internet, en tu propio ordenador, o incluso mas allá, en tu propio móvil, a través del cual puedes consultar los activos de una empresa sin necesidad de consultar su página web. Estos pequeños programas se están convirtiendo en una nueva manera de enfocar internet. Puedes crear un programa en el que consultar el tiempo que va hacer en tu ciudad, o reservar billetes de avión, o simplemente consultar la hora a través de un reloj que emplazas en el lugar del escritorio que tu elijas.

El objetivo principal de los gadget, es simplificar la finalidad del programa. Ya no necesitas crear enormes programas con muchas funcionalidades, si no que puedes crear un programa que realice un único objetivo y además puedes emplazarlo en varios lugares para compartirlo con todo el mundo.

Palabras clave: Gadget, widget, autenticación y sincronización.

Abstract

Internet, since its inception, has evolved by leaps and bounds and has allowed the society in which we live can have more information than ever before. This has led to a large extent, we can live much better and that new technologies are adapted to this giant, because that´s what really people use every day, whether at work or at your leisure.

At first everything was very simple and with very little chance, pages in which you couldn´t write anything, only the creator could edit and write at will, until it evolved to a point where we could even watch videos that could incorporate on the website that you have created by yourself. Not only that, if you arranged the knowledge to make your web page, some companies put at your disposal a new way to create your space online where you can organize this space to your linking: Blogs. And do not stay there the thing, but it also created and online company where you can have your friends to a click and chat with them in real time and upload your photos and videos to share with them and realize their opinion. And there he stayed there, but will from now you can connect to that great source of information through your phone and have it anywhere, anytime.

All this from what we just talked about is related to the social aspect, but how internet has contributed to enriching the business and employment? In an incredible way. Thanks to the internet you can send business-related documents such as charts, statistics, inventories, business plans and other documents to other departments that may be distributed throughout the world, and connect in an unbelievable way and in no time, something once thought unthinkable. We can find work thanks to the internet and send your curriculum to a huge number of companies without having to move house. Internet has helped companies and workers since the beginning of creation.

And the latest revolutionary invention that any company can make a huge profit are the gadgets and widgets that add comfort because, at this point not even need to visit the website, but you have a small program that is on the internet at you own computer, or even beyond, in your own phone, through which you can check a company's assets without consulting their website. These small programs are becoming a new approach to internet. You can create a program that will see the time to do in your city, or to reserve tickets, or just check the time through a clock that you put in your desktop.

The main objective of the gadget, the aim is to simplify the program. No need to create huge program with many features, but it can create a program that performs a single goal and you can locale it in several places to share it with everyone.

Keywords: Gadget, widget, authentication, synchronization.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción, Objetivos, Fases de Desarrollo y Estructura de la memoria.

1.1 Introducción

1.1.1 Origen y evolución de internet

Los orígenes de Internet se remontan a los años 90, como un proyecto de investigación en redes de conmutación de paquetes, dentro de un ámbito militar. A finales de los años sesenta (1969), en plena guerra fría, el Departamento de Defensa Americano llegó a la conclusión de que su sistema de comunicaciones era demasiado vulnerable. Estaba basado en la comunicación telefónica, y por tanto, en una tecnología denominada de conmutación de circuitos, que establece enlaces únicos y en número limitado entre importantes nodos o centrales, con el consiguiente riesgo de quedar aislado parte del país en caso de un ataque militar sobre esas arterias de comunicación.

Como alternativa, el citado Departamento de Defensa, a través de su Agencia de Proyectos de Investigación Avanzados (Advanced Research Projects Agency, ARPA) decidió estimular las redes de ordenadores mediante becas y ayudas a departamentos de informática de numerosas universidades y algunas empresas privadas. Esta investigación condujo a una red experimental de cuatro nodos, que arrancó en Diciembre de 1969, se denominó ARPA net. La idea central de esta red era conseguir que la información llegara a su destino aunque parte de la red estuviera destruida.

ARPA desarrolló una nueva tecnología denominada conmutación de paquetes, cuya principal característica reside en fragmentar la información, dividirla en porciones de una determinada longitud a las que se llama paquetes. Cada paquete lleva asociada una cabecera con datos referentes al destino, origen, códigos de comprobación, etc. Así, el paquete contiene información suficiente como para que se le vaya encaminando hacia su destino en

los distintos nodos que atraviese. El camino a seguir, sin embargo, no está prestablecido, de forma que si una parte de la red cae o es destruida, el flujo de paquetes será automáticamente encaminado por nodos alternativos. [1]

1.1.2 Web 1.0

La web que tuvo su nacimiento en 1989, vio la luz con su versión 1.0 que consistía en un montón de páginas en HTML (Hyper Text Markup Language) interconectados con enlaces que permitían la navegación de una página a otra. La principal característica de esta versión es que las paginas construidas son estáticas, es decir, que los usuarios no podían interactuar de ninguna manera con la pagina web, solo podían consultar la información publicada en ella. [2]

1.1.3 Web 2.0

Más tarde la evolución a las páginas web con la versión 2.0. El origen del concepto de Web 2.0 surgió en el 2004 para nombrar una conferencia organizada por O'Reilly y la empresa de soluciones de marketing Media Live Internacional, con la que querían transmitir que, lejos de haberse estrellado, la web era más importante que nunca, un espacio vibrante en el que iban y siguen surgiendo nuevas aplicaciones en línea que son abrazadas de forma inmediata y entusiasta por millones de personas en todo el mundo. O'Reilly publico en 2005 lo que hasta hoy es la principal referencia bibliográfica del concepto. Se trata del artículo "*What Is Web 2.0. Desing Patterns and Business Models for the Next Generation of Software"**. O'Reilly estableció siete principios constitutivos de las aplicaciones Web 2.0 que se resumirán a continuación:

1. *La World Wide Web como plataforma:* Salvo Linux y otras escasas alternativas, la industria del software computacional se enfrentaba casi con exclusividad a un modelo de negocio de paquetes con derechos propietarios y venta bajo el régimen de obsolescencia planificada. Dicho modelo de comercialización se enfrentará a una nueva dinámica con las aplicaciones Web 2.0. Las nuevas compañías empiezan a ofrecer software gratuito, utilizando la Web como plataforma.

2. *Aprovechar la inteligencia colectiva:* En el entorno Web 2.0 los usuarios actúan de la manera que deseen: en forma tradicional y pasiva, navegando a través de los contenidos; o en forma activa, creando y aportando sus contenidos.

3. *La gestión de la base de datos como competencia básica:* Este principio tiene una palabra clave: *infoware*: software más datos. Lo valioso de las aplicaciones Web 2.0 son los datos, ya que en muchos casos el software es un recurso abierto o de fácil implementación. Así el interés inicial de estos proyectos donde la gestión de la base de datos es la competencia básica es obtener una masa crítica de usuarios que produce un volumen de datos de gran valor.

4. *El fin del ciclo de las actualizaciones de versiones del software:* Se rompe el modelo inicial del software cerrado con derechos de uso y bajo el principio de la obsolescencia planificada, para pasar al uso del software como servicio gratuito, corriendo en la propia Web, y en combinación con los datos.

* Tim O´Reilly: What Is Web 2.0. Design Patterns and Business Models for the Next Generation of Software (*First Quarter 2007*)

5. *Modelos de programación ligera. Búsqueda de la simplicidad:* Esta noción consta en sustituir los diseños ideales de la arquitectura de la información, de los metadatos y de las interfaces gráficas por un pragmatismo que promueva a la vez simplicidad y fiabilidad para aplicaciones no centralizadas y escalables. Entre otras cosas, se pretende que las aplicaciones crezcan sin complicaciones para el desarrollador y que el usuario pueda ver los contenidos en la plataforma que desee a través de la sindicación y no cuando el desarrollador / proveedor disponga en su plataforma propietaria.

6. *El software no limitado a un solo dispositivo:* La utilización de los productos de la Web 2.0 no se limita a las computadoras. Los teléfonos móviles de tercera generación (3G) empezaron a ocupar espacios hasta ahora sólo reservados a aquellas.

7. *Experiencias enriquecedoras del usuario:* Cuando la Web era sólo contenido textual y *gifs* animados, en 1996 apareció *Flash Macromedia* para darle al usuario una experiencia más generosa a nivel gráfico. Pero la interacción de *Flash* sabe a poco con la intercreatividad y experiencia de usuario que ofrecen las aplicaciones Web 2.0 lo mismo que ocurre con los contenidos dinámicos. [3]

1.1.4 Tendencia y evolución de la Web 2.0

La Web 2.0 revolucionó internet con algunas tecnologías y tendencias inscritas en el marco de la Web 2.0 como *Rich Internet Applications (RIA,* son aplicaciones web que tienen la mayoría de las características de las aplicaciones de escritorio tradicionales*)*, *XML (*Lenguaje de marcas que permite definir la gramatica de lenguajes específicos para estructurar grandes documentos*)*, *Ajax (*técnica de desarrollo web para crear aplicaciones interactivas*)*, sindicación y agregación de contenidos en *RSS / RDF*, *Webblogs (*Ciberbitácoras*)*, *wikis (*sitio web cuyas páginas pueden ser editadas por múltiples voluntarios a través del navegador web*)*, arquitectura de información colaborativa mediante etiquetas y *folksonomias*. Asimismo se desgranan algunas posibilidades de futuro en torno a la Web 2.0, su posible evolución hacia la Web 3.0 e incluso a la *Web semántica*.

Crowdsourcing: Crowdsourcing, suele ser traducido al español como "tercerización masiva", y es un nuevo modelo de negocio a través de Internet, por el cual un trabajo concreto, como la producción de contenido, no es delegado a una persona o a una empresa, sino a un público masivo, generalmente integrado por individuos anónimos. Puede tratarse de los usuarios de un sitio web o de público en general, sometido a un mensaje propagandístico. El crowdsourcing, como operativa de trabajo, puede ser aplicado a casi cualquier área. Ya existen muchos sitios web que funcionan bajo este precepto y ofrecen sus servicios para diversas tareas, como la programación o el diseño de logos. En todos los casos, el sistema recurre a las posibilidades que permite la popularización masiva de Internet. [4]

Microblogging: Microblogging es una variación del *blog* convencional, en el cual los usuarios pueden escribir los denominados mensajes cortos -o más comúnmente denominados *posts*-, en un *blog* especial que inmediatamente se distribuye a sus amigos y a otros observadores mediante un mensaje de texto, un sistema de mensajería instantánea y por *email*. La primera aparición de los sistemas *microbloggings* fue a mediados de 2006 con el

lanzamiento de *Twitter* y se han multiplicado en otros sistemas como *Jaiku**, *Pownce*** y muchos otros. Los *posts* escritos en los *microblogging* tienen una longitud de 140 caracteres -parecidos a los mensajes de texto de los móviles- lo que reduce considerablemente el tiempo dedicado en escribir en estos sistemas de comunicación. Esto también hace posible que sea más fácil de consultar en los móviles a través de la tecnología 3G que permite consultar *microbloggings* a través de estos aparatos. [5]

Web Semántica: La web Semántica es una web extendida, dotada de mayor significado en la que cualquier usuario en Internet podrá encontrar respuestas a sus preguntas de forma más rápida y sencilla gracias a una información mejor definida. Al dotar a la web de más significado y, por lo tanto, de más semántica, se pueden obtener soluciones a problemas habituales en la búsqueda de información gracias a la utilización de una infraestructura común, mediante la cual, es posible compartir, procesar y transferir información de forma sencilla. Esta web extendida y basada en el significado, se apoya en lenguajes universales que resuelven los problemas ocasionados por una web carente de semántica en la que, en ocasiones, el acceso a la información se convierte en una tarea difícil y frustrante. Gracias a la semántica en la Web, el software es capaz de procesar su contenido, razonar con este, combinarlo y realizar deducciones lógicas para resolver problemas cotidianos automáticamente. [6]

1.1.5 GADGETS

Los gadgets llevan alrededor nuestro unos años, y los profesionales los han rechazado debido a que los veían como un juguete o una distracción, pero a lo largo de los años han ido demostrando que no lo son. De hecho, si están programados correctamente, los gadgets son una poderosa herramienta para generar una cantidad enorme de tráfico en la web. De todas formas, hay una nueva y convincente razón de por qué se merecen una consideración mucho más seria de la aplicada anteriormente: el reciente uso que han adquirido los dispositivos móviles y los smartphones para navegar por internet. Hay una sinergia natural entre estos "gadgets" físicos y sus equivalentes en la web. Al ser programas compactos y sencillos, se han convertido en la manera perfecta de entrar en este mercado emergente que no para de crecer a un ritmo vertiginoso.

El problema de definir que es un gadget, es que no existe, de momento, una estandarización de la definición formal de lo que es un gadget, Así que todo el mundo que trabaja con ellos, ha tenido la libertad de definir lo que es un gadget a su antojo, así que lo más probable es que existan diferencias entre varias definiciones.

Para el propósito de este trabajo, un gadget, es una aplicación web en miniatura que distribuye los activos de la web a la que pertenece más allá del propio sitio web. De todas formas, a pesar de ser una definición escueta, a lo largo del trabajo, se extenderá esta definición para proporcionar una más profunda y más formal de lo que realmente es un gadget, pero como comienzo, de momento nos sirve.

Una aplicación web en miniatura…

En primer lugar, un gadget, es una aplicación web. Generalmente los gadgets usan estándares web de tecnologías de desarrollo, como puede ser *XHTML*, *CSS* y *JavaScript.* A pesar de ser pequeños, los gadgets son construidos usando las mismas tecnologías de

* Jaiku: http://jaiku.com/
** Pownce: http://pownce.com/

desarrollo que cualquier otra aplicación web, de cualquier tamaño. Esto es importante, porque significa que los desarrolladores web, intrínsecamente, no necesitan aprender nada nuevo para hacer un buen uso de la tecnología gadget. Con *JavaScript* y una conexión a internet, las técnicas de *Ajax*, nos presentan una gran variedad de posibilidades de construcción. Cualquier librería existente de *CSS* y de *JavaScript* que puedas encontrar puede ser realmente útil, o incluso si se posee experiencia de desarrollo con *Flash* o *SilverLight*, estas tecnologías son también validas dentro de los gadgets.

En segundo lugar, un gadget, es una miniatura. Esto se refiere a su tamaño visual, que no ocupa más que unos cientos de píxeles de ancho y de alto, y muchas otras veces incluso más pequeño. Esto tiene evidentes implicaciones en cuanto al diseño visual: los gráficos se mantienen al mínimo, y las fuentes del texto se reducen al mínimo requerido para poder leerlo. El diseño también se ve afectado, porque para acomodarse a tan reducido tamaño, el contenido del gadget, frecuentemente, se separa en hojas o paginas discretas, a diferencia de las aplicaciones web de gran tamaño.

Más allá de la presentación, el tamaño reduce la funcionalidad hacia la esencia más pura del gadget: no pueden existir distracciones sobre el propósito principal de este. Un buen gadget solo tiene una tarea para la cual se ha optimizado. Hace un trabajo pero lo hace bien. Diseñar un gadget requiere un enfoque considerable, por que el gadget en si mismo sólo debe estar enfocado en una misión.

Finalmente, un gadget, es una aplicación web, como simple distinción respecto a una página web. Una página web de tamaño estándar, convencional, está compuesta normalmente de un cuerpo con contenidos y *links* a otras páginas, no como un gadget. Aunque debería lanzar contenido estático, este usualmente no es su rol principal. Un gadget está construido para ser autónomo, aunque eso no significa que no pueda enlazar con páginas externas, pero repito que no es su único rol.

Lo más importante, sin embargo, es que nunca son completamente estáticos. Son de naturaleza activos, entidades dinámicas, que responden a estimulaciones externas como a entradas dadas por los usuarios. Es como la diferencia entre un libro y un teléfono móvil: el libro presenta contenido estático mientras que el teléfono proporciona funcionalidad.

... que distribuye los activos de la web…

Entonces, ¿qué debería hacer un gadget? A menudo, los gadgets son parte de una larga estrategia online, que empaqueta de nuevo algunos aspectos existentes de las propiedades de la pagina web en algo de tamaño, considerablemente, más pequeño. A esto es a lo que me refiero cuando hago mención al término activo, y es la siguiente parte de la definición: ¿Qué hace a tu página web que sea más atractiva y valioso para los visitantes? ¿Cómo se puede empaquetar de manera efectiva, eso que resulta tan atractivo para el visitante, como un gadget?. Otra vez, la respuesta, no es simplemente el contenido.

... más allá de la propia página.

El último aspecto de la definición es que, a pesar de que los gadgets están basados en la web, viven fuera de la web. Esa es su razón de existencia: entregar los activos de la web, a gente que no está visitando tu página web específicamente. Esto se consigue a través de la

interfaz de programación de aplicaciones (*APIs)* que suministran las propiedades para lo que los gadgets son construidos. [7]

1.1.6 WIDGETS

Hasta hace relativamente poco tiempo atrás, el navegador era la única forma de acceder a la web. A pesar de ser fácil de usar y flexible, tiene sus limitaciones. Primero, no provee de un acceso continuo a los servicios y contenido. Para los usuarios de móviles era todo un campeonato el acceder a una página web a través de un móvil. Con los *widgets*, todas las páginas web favoritas del usuario así como las redes sociales, pueden estar presentes en el escritorio del dispositivo que se esté utilizando. El usuario puede añadir, modificar, y borrar los *widgets* a su antojo. Muchos desarrolladores se refieren a los *widgets* como un nuevo paradigma de desarrollo y despliegue de aplicaciones para los usuarios en un ambiente de computación ubicuo, donde los usuarios se mueven y usan diferentes dispositivos y redes heterogéneas.

Los *widgets* son pequeñas aplicaciones web independientes, usualmente diseñadas para una específica función individual y un acceso rápido a los servicios Web 2.0 o al contenido de internet. El World Wide Web Consortium (W3C) define los *widgets* como una conceptualización de una aplicación interactiva con un propósito único para visualizar y/o actualizar datos locales o datos en la web, empaquetados de una forma que permite una única descarga e instalación en el ordenador de un usuario o en su dispositivo móvil.

Los *widgets* deberían funcionar como una aplicación independiente, es decir, fuera de un navegador web, o deberían estar incrustados en un documento web. Al primer tipo de *widget* mencionado es clasificado como un *widget* de ordenador o de móvil dependiendo del dispositivo del que hablemos, y al segundo como web *widget.*

Los *widgets* son implementados usando el estándar de programación web del lado del cliente, así como HTML, CSS, JavaScript y XML, exceptuando si funciona en un contexto diferente. Un motor *widget*, es la colección de todos los componentes necesarios para ejecutar *widgets* en el computador del usuario o en su teléfono móvil. Esto incluye software para manejar la instalación y desinstalación de *widgets*, gestión de *widgets*, así como la interfaz funcional para invocarlo y configurarlo. Cuando un *widget* es invocado, el motor *widget* instancia el motor web, que a su vez ejecuta el *widget* en el dispositivo del usuario.

Una de las partes más cuidadas de los *widgets* de ordenador o de los dispositivos móviles es que pueden acceder a la información localizada en el dispositivo (memoria en uso, temperatura de la CPU, etc.) así como a la información localizada en la web, y mezclarlas.

El usuario puede descargar *widgets*, desde un portal de *widgets*, también conocido galerías o librerías, e instalarlos en su dispositivo. De esta forma el usuario puede customizar su ordenador o su móvil y tener acceso a cualquier tipo de información sin abrir el navegador web. [7]

1.2 Objetivos

Antes de plantear el objetivo de esta tesis me gustaría plantear una pregunta y contestarla a continuación para así poder vislumbrar el objetivo de construir un gadget o widget:

1.2.1 ¿Por qué construir un gadget / widget?

Ahora que ya sabemos exactamente lo que es un gadget y un *widget*, la siguiente pregunta obvia sería la de por qué los profesionales de la web deberían estar interesados en construirlos. A continuación expongo algunas respuestas:

La exposición gradual

La razón primaria de la construcción de un gadget es la exposición. Un gadget puede ser usado en numerosos canales de distribución lo que implica que se vea en más páginas. Incluso sin mirar más allá de la World Wide Web, los gadget tienen el potencial de expandir la visibilidad del contenido de la página web. Esto funciona ya que pueden emplazarse dentro de una página web que por sí misma recibe enormes volúmenes de tráfico. Incluso con un número pequeño de visitantes que utilicen ese gadget, el tráfico resultante puede ser sustancial.

Un ejemplo excelente es *iGoogle*, la página principal personalizada del gigante de las búsquedas y anuncios. Millones de personas visitan esta página cada día, lo que significa que no necesitas tener un gadget muy popular para que los usuarios lo utilicen. En la figura 1.1 vemos un informe analítico de las cientos de miles de visitas a un gadget medianamente popular en apenas 3 meses de lanzamiento. Hay que tener en mente que es solo un gadget en un solo portal. ¿Qué podrá conseguir publicado en más de una?

Figura 1. 1 Visitas a un gadget

Gracias a la portabilidad con la que podemos caracterizar a un gadget, se nos brinda la posibilidad de expandirlo más lejos de la web, como puede ser incorporarlo en el escritorio

del ordenador o en dispositivos portátiles. Los gadgets proporcionan una oportunidad única de influir en los activos de la página y en las técnicas de desarrollo más allá de la web. Ninguna otra tecnología de la web permite tanta extensión de esta manera.

El segmento de los smartphones se puede aprovechar de este enorme potencial. A principios de 2009 se estimó que más de 4 billones de teléfonos móviles navegarían por internet, y estarían configurados para soportar gadgets, lo que representa un porcentaje creciente en este mercado.

Los beneficios de una exposición gradual

El beneficio más obvio es el tráfico que puede generar un gadget. El objetivo es conseguir que el gadget muestre los activos de tu página web, porque así, a través de un programa pequeño cumplirás con tus objetivos, sin necesidad de que utilicen tu página web.

Por ejemplo, considerar un gadget producido por una agencia de viajes que proporciona una interfaz a su motor de búsquedas. Cuanto más lejos consigas colocarlo, más probable será que los usuarios lo utilicen antes que meterse en la página web. Por lo tanto, los de la agencia conseguirán que más gente utilice su motor de búsqueda para consultar viajes, y que si es bueno, lo sigan utilizando para futuros viajes. Así es que estarás desplazando los activos de la web mucho más lejos de tu propia página y conseguirás aumentar el tráfico.

El pequeño tamaño del gadget influye en que, generalmente, no se puede proveer todas las funcionalidades que la página web puede ofrecer. Para acceder a las funcionalidades que no entran dentro del gadget los usuarios deberán meterse en la página web, y un gadget en su escritorio podría significar un ruta directa hacia esas funcionalidades sin necesidad de abrir el navegador y escribir la ruta hacia tu página sino que en el escritorio de su ordenador puede tener un enlace directo a la página. En este sentido el tráfico no se limita al gadget en sí mismo.

Volviendo al ejemplo de la agencia de viajes, imagina que el motor de búsqueda no encaja dentro del gadget. En este caso, una alternativa viable podría ser la de mostrar, en el gadget, ofertas actuales de tarifas de vuelos, o las tendencias de vuelos de los usuarios, con un enlace al sitio web principal para que pueda, una vez consultado estas opciones, meterse dentro del motor de búsqueda y elegir un viaje, teniendo una idea de viaje, según lo que ya le has mostrado a través del gadget. Los usuarios, por tanto, son alentados a que se les enlace a la página cuando ven un billete de su interés.

Finalmente, en algunos casos, el tráfico que el gadget puede generar se basa en sí mismo, retroalimentándose para lograr un crecimiento aun mayor. El termino *marketing viral*, aunque a veces se usa en exceso, se puede aplicar en este caso: el medio del gadget es el mensaje, y el mismo acto de utilizarlo promueve ese mensaje.

Otro beneficio de desarrollar gadgets proviene de su tamaño reducido. A pesar de que el tráfico que pueden generar es enorme, el esfuerzo de generar el código no lo es, dando un excelente rendimiento en la inversión de su desarrollo. Es raro que un gadget necesite más de unos cuantos cientos de líneas de código. Esto es especialmente cierto si además el gadget utiliza algún paquete de alguna aplicación existente; puede ser capaz de reutilizar algunos servicios web o librerías que se utilicen en el sitio web. Pero incluso si se tiene que desarrollar desde cero, se tiene que tener claro, como se comentó en una parte anterior del trabajo, que el gadget tiene que hacer “una cosa” pero la tiene que hacer bien, esto implica

que es menor la proliferación de código, simplemente porque hay menos características que implementar.

Un beneficio adicional de tener pocas líneas de código es que no necesitas a un gran equipo de programadores para desarrollarlo. Esto hace que sean asequibles para los desarrolladores o para empresarios individuales. Este hecho hace que sea más atractivo para las grandes corporaciones; no solo es más fácil encontrar los recursos de desarrollo, sino que también es más eficiente si el equipo de desarrollo es más pequeño, cosa que convierte al gadget en algo bastante óptimo en cuanto a su desarrollo.

Una vez que el gadget está desplegado, los beneficios de su código base tan pequeño, continúan refiriéndonos en términos de mantenimiento. Al ser tan pequeños los errores son mucho más fáciles de encontrar así como es mucho más difíciles que se produzcan.

Una vez que hemos comprobado cual es el objetivo de construir un gadget/widget vamos a responder a la pregunta de cuál es el objetivo de este trabajo. [7]

1.2.2 Objetivo del Trabajo

El objetivo de este trabajo no es más que la construcción de una pequeña aplicación que sirva para cumplir una única meta que simplifique la vida de la gente que sienta la necesidad de usarlo. Y no es otro que poder sincronizar las tareas diarias de un trabajador común a través de dos calendarios de enorme uso (Microsoft Outlook y Google Calendar) en la vida laboral de cualquier trabajador que utilice un ordenador y así poder disponer de su lista de tareas ya no solo en su ordenador de trabajo sino también en la nube para poder consultarla aunque no este en su puesto de trabajo y poder agregar nuevas tareas, y más tarde, cuando vuelva al trabajo pueda actualizar su lista de tareas.

Resulta que la empresa Google lanzó un gadget que sincronizaba los eventos que podías incluir en su calendario y a través de esta aplicación podías sincronizarlos con el calendario de Microsoft Outlook y viceversa. El problema es que se quedaron ahí, simplemente con los eventos, y se olvidaron de la parte de las tareas.

Y dado que Google consideró que este pequeño gadget cumplía una función básica, he considerado que se les olvido terminar su trabajo por lo que ofrezco mi pequeño grano de arena para que las tareas de trabajador se simplifiquen de alguna manera y no pierda el tiempo en copiar a mano sus tareas de un calendario a otro, aspecto que es demasiado trivial y por el que no deberían perder nada de tiempo.

En base a este objetivo principal, se proponen los siguientes objetivos parciales:

- La autenticación respecto de Google se realizará mediante el protocolo OAuth2 por el cual se tendrá que acceder a la cuenta asociada de Gmail y permitir el acceso a los datos para posteriormente dotarle de un token de acceso a esos mismos datos.

- Se guardará el token proporcionado por Google para no tener que estar autenticando constantemente.

- No existe autenticación respecto de Outlook a no ser que este configurado de otro modo, y sí existirá la autorización para coger los datos necesarios de Outlook.

- Los usuarios podrán crear, modificar y eliminar una tarea que este en ambos calendarios y quedara reflejado en el otro.

- Se realizará una pequeña base de datos para mantener la consistencia con las tareas creadas y eliminadas.

- El gadget, como hemos visto debe ser pequeño en cuanto a gráficos, por lo tanto se dotará de una interfaz de usuario simple y comprensible para todos sus usuarios.

- Se minimizará a la barra de tareas para ser lo menos molesto posible.

- Contara con dos opciones de sincronización: una para configurar y que realice la sincronización cada X minutos elegidos por el usuario, y otra, más inmediata, que sincroniza cuando el usuario lo elija.

- Existirán 3 opciones de sincronización: De Outlook a Google, de Google a Outlook, y los dos a la vez.

1.3 Fases de desarrollo

Fase 1: Elección de la aplicación. Estudio sobre el tipo de gadget que se quiere realizar: Web, para telefonía móvil o aplicación de escritorio.

Fase 2: Estudio de la aplicación Outlook. Estudio sobre el programa Microsoft Outlook para saber como abordarlo y cual es la mejor manera de acceder a los datos contenidos en el.

- **Fase 2.1: Alternativa Libpst.** Mediante un archivo con extensión .PST individual de cada cuenta de correo asociada a un solo usuario. Mediante API libpst.
- **Fase 2.2: Alternativa Jacob.** Mediante el archivo global con extensión .PST general que asocia todas las cuentas de un solo usuario. Mediante API Jacob.

Fase 3: Estudio de la aplicación de Google. Estudio sobre Google Calendar para saber como abordarlo y cual es la mejor manera de acceder a los datos contenidos en el.

- **Fase 3.1: Estudio Autenticación.** Estudio de las diferentes formas de autenticación que dispone Google para poder acceder a los datos de usuario.
  - Ventajas e inconvenientes de protocolo Oauth 2.
  - Ventajas e inconvenientes de protocolo Oauth 1.
  - Ventajas e inconvenientes de protocolo OpenID.
  - Ventajas e inconvenientes de protocolo AuthSub.
  - Ventajas e inconvenientes de protocolo ClientLogin.
- **Fase 3.2: Estudio Acceso Tareas.** Estudio de API Google Task para poder acceder a las tareas contenidas en el calendario de Google.

Fase 4: Implementación de la aplicación. Esta fase comprende todo el desarrollo de la aplicación que se divide en estas subfases:

- **Fase 4.1: Acceso a las tareas de Outlook.** Se programa el acceso a las tareas que están almacenadas en el calendario de Outlook.
- **Fase 4.2: Autenticación.** Se realiza la autenticación para poder acceder a los datos de Google. Se guarda el token para posteriores usos.
- **Fase 4.3: Acceso a las tareas de Google.** Se programa el acceso a las tareas que están almacenadas en el calendario de Google.
- **Fase 4.4: Sincronización.** Se realiza la sincronización en la dirección elegida por el usuario.

- **Fase 4.5: Tiempo sincronización.** Se realiza la tarea de sincronizar cada X minutos.

Fase 5: Creación de la GUI. Se desarrolla la interfaz grafica de usuario con todos los botones y opciones de pantalla, y se subdivide en estas fases:

- **Fase 5.1: Programación de las opciones.** Se programa las diferentes opciones de sincronización.
- **Fase 5.2: Programación botón Autenticación.** Se une la fase 4.2 con el botón que le da paso.
- **Fase 5.3: Programación botón Sincronización.** Se une la fase 4.4 con el botón que le da paso.
- **Fase 5.4: Programación tiempo sincronización.** Se une la fase 4.5 para dar a elegir al usuario el tiempo de sincronización.
- **Fase 5.5: Programación minimizar.** Se programa la opción de minimizar el programa a la barra de sistema.
- **Fase 5.6: Programación panel barra sistema.** Se divide en:
    - **Fase 5.6.1: Programación botón Sincronizar.** Se programa la opción de sincronizar al instante.
    - **Fase 5.6.2: Programación botón Restaurar.** Se programa la opción de restaurar la aplicación al escritorio de nuevo.
    - **Fase 5.6.3: Programación botón Salir.** Se programa el botón de salir del programa.

Fase 6: Realización del .jar. Creación del ejecutable del programa.

Fase 7: Realización del instalador. Se crea el instalador de la aplicación.

Fase 8: Documentación. Elaboración de una memoria en la que se explica con detalle la finalidad de la aplicación, se describen las tecnologías empleadas y los requisitos necesarios para llevarla a cabo.

Figura 1. 2 Diagrama Gantt

1.4 Estructura del documento

Para facilitar la lectura de la memoria, se incluye a continuación un breve resumen de cada capítulo:

- **Capítulo 2**
    - **Plataformas:** Las plataformas disponibles para hacer correr los gadgets.
    - **Estado del Arte:** Estudio detallado de las posibles tecnologías a utilizar y las que se utilizan para la realización del proyecto.
    - **Visión de alto nivel:** Todos los elementos software y hardware para poder llevar a cabo el proyecto
- **Capítulo 3**
    - **Presupuesto:** Cálculo del presupuesto necesario para el desarrollo de la aplicación, incluye tanto costes materiales como de personal.
- **Capítulo 4**
    - **Conclusiones:** Exposición sobre las conclusiones generales obtenidas tras la implementación del proyecto.
    - **Trabajos futuros:** posibles trabajos futuros propuestos para contribuir a la mejora del proyecto desarrollado.

Capítulo 2

Plataformas, estado del arte, visión de alto nivel y estudio UML.

2.1 Plataformas

Los gadgets empezaron en la web, y ese sigue siendo su hábitat natural, aunque se han extendido más lejos de ese hábitat. Usando un API de un navegador, tu gadget puede ser colocado en la página de cualquier otra persona dando a los visitantes de esa página acceso directo a los activos de tu página web. Esto proporciona beneficios para todas las partes implicadas. Para el propietario del gadget, ya que amplía la distribución de su marca y de su contenido, aumentando su exposición. Para el propietario de la página que lo contiene, ya que así está enriqueciendo su propia página con nuevas funcionalidades; añade módulos de contenido nuevos con el mínimo esfuerzo. Y para los usuarios que visiten esa página ya que están enriqueciendo su propia experiencia al navegar por internet.

Por ejemplo una página que promocione el turismo, y que realice visitas a un lugar, puede añadir una gadget que muestre el tiempo que va a hacer en esa localidad mientras dura su estancia en ese lugar. En ese proceso la página web del tiempo, de donde viene el gadget se beneficiara de ello ya que, a lo mejor, la próxima vez que quiera consultar el tiempo, recuerda que existe una página que posee una aplicación de fácil uso para consultar el tiempo en otro lugar.

Todo esto ha derivado en una conclusión lógica, crear un lugar donde cada usuario pueda agrupar los gadgets como más le guste y elegir los que le apetezca. Notar que en este caso es el usuario el que elige las aplicaciones que quiere tener en su página principal y no el webmaster de la página, que otorga al usuario la posibilidad de tener una lealtad a la página web. Un par de ejemplos de páginas web que realicen esto son: iGoogle* y Netvibes**.

* iGoogle: http://www.google.es/ig
** Netvibes: http://www.netvibes.com/es

2.1.1 Ordenadores

Una parte de lo que hace que un gadget sea tan atractivo es que son aplicaciones web que pueden ser utilizadas fuera de un navegador web. Esto es posible gracias a las APIs que hacen funcionar los gadgets web directamente en los Mac OS x y en Windows Vista/XP/7. Simplemente se instala el gadget como un paquete independiente en el ordenador sin la necesidad de utilizar un navegador web. Esto permite un fácil acceso al contenido y a la funcionalidad solo con que el ordenador este encendido. De esta manera se tiene un alcance mayor con el usuario desplazando el gadget más lejos de la navegación por internet y consiguiendo que acceda a los activos de tu página web sin ni siquiera conectarse a internet.

El soporte de estos gadgets proviene de los dos grandes sistemas operativos del mercado, como también de terceras partes. Estos gadgets son un poco más complejos que sus hermanos en la web, requieren alguna interfaz más, pero los principios son básicamente los mismos.

2.1.2 Dispositivos Móviles

Finalmente, los gadgets tienen el poder de desplazarse más lejos aún del ordenador y de internet y aterrizar justo en la palma de la mano, dispuestos a ir contigo a cualquier lugar donde vayas. Esto es posible gracias a la nueva generación de móviles que existen en el mercado actual de telefonía. Estos ordenadores de mano tienen la capacidad de ejecutar aplicaciones web en cualquier lugar y los gadgets son el medio natural para conseguirlo.

Técnicas de Distribución

Hay dos principales maneras de distribuir los gadget en los teléfonos móviles:

- ***Navegador del móvil*:** La manera más inmediata que tienen los teléfonos móviles de hacer funcionar un gadget es a través de sus propios navegadores. Como norma estos dispositivos incluyen un navegador web preparado especialmente para ellos, capaces de ejecutar casi cualquier aplicación web que exista hasta el momento, y si no es así, cada vez más se están adecuando los programas para poder funcionar en estos dispositivos.

- ***Aplicación para móvil*:** Existen también una variedad de técnicas y de APIs para empaquetar las aplicaciones web y convertirlas directamente en aplicaciones de estos dispositivos tan novedosos. Como en los gadgets para ordenadores, esto permite a la aplicación ser instalada directamente en el móvil sin necesidad de estar conectado a internet.

2.1.3 Web distribuida

Los canales descritos anteriormente comprenden la participación de los gadgets web en lo que a veces se llama web distribuida. Es una evolución natural en el desarrollo web, donde el contenido web y la funcionalidad no están confinados en los tradicionales sitios web.

La web distribuida comenzó con la popularización de los archivos *RSS*. Como estos archivos comenzaron a tener un uso generalizado, no fue necesario visitar el sitio web para

ver las actualizaciones. De hecho, los agregadores de estos archivos *RSS* permitieron a los usuarios coleccionar una gran variedad de *feeds* que les interesaban y poder visitarlos en un solo lugar. En el proceso, los proveedores de contenidos vieron un incremento muy marcado y su potencial creciente; porque la gente podía acceder a su información con un esfuerzo menor, esto contribuyó a que más gente lo utilizase.

Los gadgets web son, considerablemente, más poderosos que los simples *feeds*, pero compartían la ventaja de que los usuarios podían consultarlos sin necesidad de visitar el sitio de donde provenían constantemente. Y como con los RSS feeds, los usuarios de los gadgets ganan el poder de seleccionar exactamente el contenido que quieren, mejorando el control de la experiencia de navegación diaria.

Otros dos aspectos relacionados con la web distribuida son la proliferación de APIs y el consecuente incremento de los *mashup* web (Es una aplicación o página web, que usa y combina el contenido, la presentación o la funcionalidad desde dos o más recursos para crear un nuevo servicio). Este fenómeno proporciona a los desarrolladores web amateurs la habilidad de combinar funcionalidad desde diferentes sitios en nuevos e interesantes caminos, incluso crear nuevos servicios que son más que la suma de sus partes. Similarmente, los gadgets web pueden ser vistos como un análogo de un API, pero para usuarios finales más que para los desarrolladores. Usuarios son capaces de mezclar contenido y funcionalidad desde una gran variedad de fuentes, creando con precisión la página principal. Otra vez los usuarios tienen el control sobre su experiencia informática.

2.1.4 Compatibilidad entre plataformas

Los gadgets son probablemente únicos en cuanto al número de diferentes plataformas para los que se pueden construir -web, en el ordenador y en dispositivos móviles donde se desarrollan con un código base más o menos simple. Podemos pensar en los estándares de *XHTML*, *CSS* y *JavaScript* para desarrollarlos, sin embargo ni siquiera los sitios web donde nacieron estas tecnologías pueden hacer salir programas de la manera en que lo hacen los gadgets. En el código base, corregir los errores y realizar las mejoras seguramente solo sea necesario, hacerlas en un sola vez para todas las plataformas donde se ejecutan los gadgets.

Este beneficio destaca, sobretodo, en los smartphones. Dado que ningún sistema operativo para móviles se ha establecido como el líder indiscutible en todas partes, el desarrollo de una aplicación nativa para cualquier plataforma abrirá sólo un pequeño segmento del mercado de los móviles. Las tecnologías web que soportan los nuevos dispositivos móviles están bien estandarizadas, así que el mismo código base se puede utilizar para todos los dispositivos móviles que soporten la web, incluso los que todavía no existen.

2.1.4.1 Estrategias de compatibilidad entre plataformas

No obstante, las ventajas de la compatibilidad entre plataformas no son del todo intuitivas. Las diferentes APIs son suministradas por los proveedores de servicios, cada API requiere su propio esfuerzo de desarrollo. A continuación se presentaran dos posibles estrategias para que nuestro gadget llegue lo más lejos posible, y sea compatible con el máximo de plataformas, para así poder expandirnos más.

Opción 1: *Autónomo*

La técnica mas obvia para la compatibilidad entre plataformas es mantener el gadget autónomo, sin usar ningún tipo de *API*. En otras palabras, desde que todas las plataformas soportan las tecnologías web básicas, como *XHTML*, *CSS* o *JavaScript*, cualquier gadget que se adhiere estrictamente a estos estándares, y no usa nada más, funcionará en cualquier lado. A este acercamiento se le llama "autónomo" por que funciona mejor para gadgets que utilizan algunos recursos externos. En particular, los gadgets se adaptan bien a esta técnica si su funcionalidad está integrada internamente (así como en *Flash* o en *JavaScript*) en lugar de requerir la interacción con el servidor.

Cuando esta técnica es posible, es la mejor que se puede utilizar. Sin embargo, los requisitos del mundo real a menudo dictan una arquitectura más interconectada que se puede lograr de esta manera.

Opción 2: *Una capa intermedia*

Debido que las APIs de los diferentes gadgets están cumpliendo un rol muy parecido, comparten una funcionalidad similar. Fundamentalmente, estas se orientan hacia la ampliación de las capacidades genéricas que se encuentran en *JavaScript* de manera que son especialmente útiles para los gadgets. Específicamente, muchas de las APIs contienen funciones en las siguientes áreas:

- ***Preferencia al cargar y salvar:*** Es un requerimiento común para los gadgets guardar la configuración de una sesión a la siguiente, incluso en las plataformas de gadgets como puede ser iGoogle, y que se mantengan los ajustes de una plataforma a otra para no tener que estar configurándolos constantemente.

- ***Recuperación de contenido:*** Esto incluye la búsqueda de datos de texto sin formato, aplicaciones *XML* para *Ajax*, y ocasionalmente imágenes.

- ***Funciones relacionadas con el tamaño:*** La pantalla tan pequeña de los gadgets hace que el tamaño se convierta en algo a tener en cuenta.

Debido a que estas funciones son muy similares entre una API y otra, es posible crear una librería concisa de *JavaScript* que abstraiga las funciones en un sola pseudo-*API*, que aisle el código del gadget de las dependencias de cualquier plataforma en particular.

Una vez establecido el planteamiento del problema vamos a ver que relación tiene con este proyecto. Para este proyecto en concreto voy a elegir hacer una aplicación de escritorio, ya que de momento no existe el Outlook para móvil, y menos para Android que seria la plataforma que escogería en el caso de elegir una de las disponibles hoy en día para el mercado, pero eso es otro tema.

La aplicación web requeriría el alojamiento en alguna página web o en alguna plataforma para gadgets de las que existen en el mercado, puede ser un trabajo futuro pero de momento considero que el mejor emplazamiento para el gadget a realizar es el escritorio del ordenador.

En cuanto a la compatibilidad entre plataformas, se va a construir el gadget para Windows ya que uno de los calendarios escogidos es el de Microsoft Outlook ya que la gran mayoría de empresas trabajan con Windows y en concreto para los emails también esta muy difundido el Outlook.

Y en cuanto a la estrategia de compatibilidad, definidas anteriormente, se va a escoger la opción 2 de capa intermedia ya que encajan sus 3 preceptos perfectamente con nuestro pequeño programa. Por un lado se guarda el token de acceso para que no se tenga que estar autenticando constantemente el usuario cada vez que quiera utilizar el programa. Se establece una base de datos para mantener la consistencia de las tareas para comprobar que tareas se han creado y eliminado desde la última vez que se realizó una sincronización. Y por ultimo, el gadget posee una interfaz muy simple y de reducido tamaño para que suponga una utilización fácil para cualquier tipo de usuario que haga uso de él y así cumpla con las directrices, mencionadas anteriormente, que debe cumplir un gadget. [7]

2.2 Estado del arte

2.2.1 Diferentes plataformas de construcción

2.2.1.1 Plataformas para novatos

En estas plataformas se ha simplificado al máximo la creación de gadgets, partiendo de plantillas y sin apenas tener que generar código, lo único que tienes que hacer es pasar por varias etapas escogiendo el aspecto que quieres que tenga el gadget (tamaño, colores, ancho, alto, patrones...). Se genera sin que tengas que escribir código y sin hacer pruebas. Según la página web donde lo construyas, se escribe en *HTML*, *Flash*, *JavaScript*. Por supuesto, la principal desventaja de esto es que no tienes la absoluta libertad para hacerlo como quieras, si no que tienes la libertad de elegir opciones de construcción a partir de plantillas predefinidas. La ventaja es que es muy fácil de construir, sobre todo para alguien que no posee nociones de programación. A continuación se listan unos cuantos ejemplos de páginas donde se construyen los gadgets:

2.2.1.1.1 Widgetbox

Figura 2. 1 Logo Widgetbox

Widgetbox es un servicio en línea para crear diversos tipos de widgets para distribuir su contenido en la web. Puede crear widgets en *Flash* o *HTML*, Facebook Apps, Google Gadgets, widgets, Twitter, y más. Para crear y personalizar el widget, puedes utilizar uno de los modelos disponibles en el sitio. Cada widget creado con Widgetbox es redistribuible plenamente en los principales sitios de intercambio social, incluso el widget en sí. Ningún análisis de los recursos está disponible en el plan gratuito, pero si desea realizar un seguimiento del rendimiento de su widget (y no optar por ver los anuncios de manera gratuita) existe una posibilidad de suscripción mediante pago.

2.2.1.1.2 KickApps

Figura 2. 2 Logo KickApps

KickApps es un servicio gratuito en línea donde se puede crear widgets para distribuir tu contenido y promover tu marca en la web. KickApps genera widgets en Flash o HTML desde cero o usando uno de los modelos disponibles. Cuando el widget esté incrustado en tu sitio web, los lectores pueden redistribuirlo utilizando sitios de social media. También puedes crear un WidgeADs, que es un tipo especial de widgets que sirve como un anuncio y puedes

distribuirlo de forma gratuita para ayudar a redirigir el tráfico de vuelta a tu sitio web. El registro es necesario para utilizar el servicio. Ofrece características de monitoreo del rendimiento de tu sitio web (análisis).

2.2.1.1.3 SpinletsLab

Figura 2. 3 Logo SpinletsLab

SpinletsLab es un servicio gratuito online para ayudar a crear, distribuir y administrar tus widgets. Puedes crear widgets en *Flash* desde cero o usando uno de los modelos disponibles. Una vez creado el widget, puedes distribuirlo a todos los sitios de social media e incluso en teléfonos móviles. Tus lectores pueden difundir tu contenido a través de un botón dedicado para el compartimiento del widget. Por último, puedes rastrear su acción en tiempo real, utilizando la función de análisis. El registro es necesario para utilizar SpinletsLabs.

2.2.1.1.4 Produle

Figura 2. 4 Logo Produle

Produle es un servicio para la creación de *widgets* que puede generar tres *widgets* basados en *Flash*, sin ningún costo. Tienes varios modelos a disposición y también un estudio basado en la web para añadir contenido multimedia como videos, imágenes o música con tan sólo unos pocos clicks. Puedes utilizar Produle con libertad para distribuir tu contenido en los sitios de intercambio social e incluso para que tus lectores propaguen tu contenido de modo viral a otros sitios usando la función de distribución social disponible en cada elemento. El registro es necesario para utilizar el servicio.

2.2.1.1.5 Widgadget

Figura 2. 5 Logo Widgadget

Widgadget es un creador de *widgets* gratuito que puedes utilizar para crear *widgets* y distribuir contenidos en sitios Web y medios de comunicación social. Puede crear widgets en *Flash* o *HTML* a partir de uno de los modelos disponibles. Estos modelos están organizados para ayudar a elegir el mejor *widget* para tu público objetivo (bloggers, usuarios de escritorio, etc.). Ofrece análisis de rendimiento y recursos para el intercambio social. Widgadget requiere el registro.

2.2.1.1.6 Grazr

Figura 2. 6 Logo Grazr

Grazr es un servicio en línea gratuito que puedes utilizar para crear widgets que contengan canales RSS. Los feeds *RSS* pueden contener texto, imágenes, música o vídeos. Para subir varios feeds RSS al mismo tiempo, el servicio soporta la importación de archivos *OPML*. Puedes elegir entre tres modelos hechos para dar estilo a tu widget, que proporcionan características para el análisis, o recursos para compartir en sitios de medios sociales. El registro en Grazr es completamente gratis.

Dado que estas plataformas son muy triviales no se va a ofrecer una comparativa entre ellas. [8]

2.2.1.2 Plataformas para profesionales

A diferencia de las plataformas anteriores, en estas sí que se requieren ciertas nociones básicas de programación orientada a la web. No implica que sea una programación complicada, pero sí que se requieren ciertos conocimientos de *CSS*, *XML*, *HTML*... Algunas de ellas simplemente están más orientadas a profesionales que sepan lo que quieren y que tengan cierta idea de lo que pretenden hacer. Para poder hacer un gadget, generalmente se tendrá que leer la documentación proporcionada por la empresa que facilita el programa de creación de gadgets. La principal ventaja es que tienes toda la libertad que quieras para poder realizar tu gadget. La desventaja es que tienes que tener una buena base de programación orientada a la web. A continuación se muestran unos ejemplos:

2.2.1.2.1 Yahoo Widgets

Figura 2. 7 Logo Yahoo Widgets

En realidad es una plataforma donde puedes escoger los widgets que más te gusten e integrarlos en tu ordenador con solo descargarlos. Pero ofrece la posibilidad de construir widgets e insertarlos en Konfabulator (Konfabulator es una aplicación que permite insertar Widgets para poder usarlos). Gracias a esta aplicación podremos crear cualquier widgets imaginable con unas nociones de *CSS*, *JavaScript* y *XML*. También proporcionan algunas herramientas para facilitar la creación del widget. [9]

2.2.1.2.2 Google Gadgets

Figura 2. 8 Logo Google Gadgets

Google Gadgets viene con Google Desktop. Al igual que el anterior es una plataforma que te permite integrar los gadgets que quieras al Google Desktop. Y si no a través de internet con iGoogle. También tienes la posibilidad de crear cualquier gadget gracias al API de gadgets, que sin ningún programa añadido que tengas que descargar te proporciona las herramientas básicas para construir cualquier gadget. Utiliza *JavaScript*, *XML* y *CSS*. [10]

2.2.1.2.3 Adobe Air

Figura 2. 9 Logo Adobe Air

El tiempo de ejecución de Adobe AIR permite a los desarrolladores utilizar *HTML*, *JavaScript*, *Adobe Flash Professional* y *ActionScript* para crear aplicaciones web que se ejecutan como aplicaciones clientes independientes sin las restricciones de un explorador. Adobe AIR, un componente clave de la plataforma *Flash*, da rienda suelta a la creatividad de los diseñadores y desarrolladores al ofrecer un entorno de desarrollo flexible y coherente para la creación de aplicaciones en dispositivos y plataformas. [11]

2.2.1.2.4 Google Web Toolkit

Figura 2. 10 Logo Google Web Toolkit

En realidad no es un constructor gadget como tal sino un creador de aplicaciones web. ¿Pero que es un gadget si no una aplicación web en miniatura? La creación, la reutilización y el mantenimiento de una gran cantidad de componentes *AJAX* y bases de código *JavaScript* pueden ser tareas complejas y delicadas. Google Web Toolkit (GWT), especialmente en combinación con el complemento de Google para Eclipse, facilita estas arduas tareas al ofrecer a los desarrolladores la posibilidad de crear y mantener rápidamente aplicaciones *JavaScript* con interfaces complejas, pero de gran rendimiento, en el lenguaje de programación Java. [12]

2.2.1.2.5 Windows Sidebar

Figura 2. 11 Logo Windows Sidebar

Motor Widget que se instala en el escritorio de tu ordenador y te permite tener una lista de widgets, puedes agregarlos o borrarlos según tus necesidades. Consiste en una barra lateral en tu escritorio en el que puedes acumular los widgets que más utilices. Para crearlos no necesitas ningún programa, con el bloc de notas y conocimientos en *HTML*, *XML* y *JavaScript*, podrás crearlos. En la página web de Windows te proporcionan una gran lista de widgets para descargar e incorporarlos en tu Sidebar. [13]

2.2.1.2.6 Mac Dashboard Widget

Figura 2. 12 Logo Mac Dashboard Widget

Al igual que el anterior es un motor widget para Mac que te permite tener en el escritorio de tu Mac una lista de widgets. Desde la página web de Mac, te ofrecen una larga lista de widgets para descargar en tu Mac y con una simple instalación, puedas empezar a utilizarlos. Para la creación de widgets te ofrecen, con el sistema operativo *Snow Leopard*, un programa llamado *Xcode* con el que puedes empezar a construir widgets. Para el aspecto visual de tu widget te proporcionan *Interface Building* con el podrás diseñar el aspecto que quieres que tenga tu widget y personalizarlo según tus gustos. [14]

Realicé un estudio previo a este proyecto en el que se realizaba una comparativa y se evaluaban estas tecnologías en base a unas características y se estimaba cuál era la mejor. No considero oportuno incluir el estudio entero pero si las dos tablas resumen para que a simple vista se pueda sacar una conclusión.

	Tecnologías Soportadas	Tecnologías Especificas	Sistemas Operativos	Soportes y Herramientas	Librerias y Frameworks	Licencias de Desarrollo	Curva Aprendizaje
YahooWidgets	XML CSS JAVASCRIPT	YahooWidgets (Konfabulator)	Mac OSX Windows	Widget Converter Widget Class Library, UNIX Utilities for Windows, Más...	Librería YUI	No	Media*
Google Gadgets	XML CSS JAVASCRIPT	Google Desktop, iGoogle	Mac OSX Windows	Diseñadores Gadgets Google Desktop	API Google	No	Baja*
Adobe Air	HTML JAVASCRIPT ACTIONSCRIPT 3 FLASH	Adobe AIR	Mac OSX Windows Linux	Flash Professional y Builder Dreamweaver	API Adobe	No	Media*
GWT	JAVA XML JSNI	Google App Engine, Google Desktop	Mac OSX Windows Linux	Eclipse SDK Speed Tracer GWT Designer	API Google	No	Baja*
Windows Sidebar	XML HTML JAVASCRIPT	Silverlight Sidebar	Mac OSX Windows Linux	Silverlight	API Silverlight	No	Alta*
Mac Dashboard	HTML CSS JAVASCRIPT	Widgets Dashboard	Mac OSX	Xcode IDE, Interface Builder, Análisis Funciones	Core Data	No	Baja*

Tabla 2. 1 Comparación I

* Alta: Aprendes en mucho tiempo. Media: Aprendes en un tiempo moderado. Baja: Aprendes en poco tiempo.

	Documentación Aportada	Ejemplos	Plantillas	Facilidad Desarrollo	Mecanismos Soporte	Idioma	Universalidad	Tiempo Creación
YahooWidgets	**Alta****	**Bajo**	**No**	**Media**	**Foro Manuales**	**Ingles Español Australiano Japonés Chino**	**YahooWidgets**	**Medio**
Google Gadgets	**Alta****	**Alto**	**No**	**Alta**	**Foro Manuales Blog**	**Ingles Español Chino Portugués Ruso**	**iGoogle Google Desktop Blogger Orkut Más...**	**Bajo**
Adobe Air	**Media****	**Medio**	**No**	**Media**	**Foro**	**Ingles Español**	**Dispositivos Móviles y TV Ordenador**	**Medio**
GWT	**Alta****	**Alto**	**No**	**Alta**	**Foro Manuales Blog**	**Ingles Español Chino Portugués Ruso**	**Google App Engine, Google Desktop**	**Bajo**
Windows Sidebar	**Baja****	**Bajo**	**No**	**Baja**	**Blog Foro**	**Ingles Español**	**Windows Live, Messenger Connect**	**Alto**
Mac Dashboard	**Media****	**Bajo**	**Si**	**Media-Baja**	**Foro de pago Manuales**	**Ingles**	**Dashboard iPhone**	**Alto**

Tabla 2. 2 Comparación II

** El análisis que se utiliza en Documentación Aportada se basa en una relación entre la facilidad de encontrarla, la presentación y el contenido.

2.2.1.3 Plataformas escogidas

2.2.1.3.1 Java

Java es un lenguaje de programación muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general. Fue desarrollado por Sun Microsystems y está enfocado a cubrir las necesidades tecnológicas.

Una de las principales características es que es un lenguaje independiente de la plataforma, es decir, un programa en Java podrá funcionar en cualquier ordenador indistintamente del sistema operativo. Es una ventaja significativa para los desarrolladores de software, pues antes tenían que hacer un programa para cada sistema operativo, por ejemplo Windows, Linux, Apple, etc. Esto lo consigue porque se ha creado una Máquina de Java para cada sistema que hace de puente entre el sistema operativo y el programa de Java y posibilita que este último se entienda perfectamente. Otra de las ventajas es que Java está desarrollándose para distintos tipos de dispositivos además del ordenador como móviles, agendas y en general para cualquier cosa que se le ocurra a la industria.

La interoperatividad que proporciona la plataforma Java es también muy útil para los usuarios de dispositivos móviles. Las aplicaciones creadas con los APIs estándar de Java deben ejecutarse en todos los dispositivos compatibles, no importa quien los fabricó.

Existen básicamente tres plataformas:

- Java Platform, Standard Edition (Java SE).
- Java Platform, Enterprise Edition (Java EE).
- Java Platform, Micro Edition (Java ME). [15]

2.2.1.3.2 Google APIs Client Library para Java

Escrita por Google, esta biblioteca es una biblioteca Java de cliente flexible, eficiente y poderosa para acceder a cualquier APIHTTP basada en la web. Cuenta con un protocolo potente y fácil de usar llamado OAuth 2.0 con una interfaz consistente.XML y JSON son los modelos de datos elegidos, de peso ligero, eficiente, y para apoyar cualquier esquema de datos. [16]

En su interior se compone de las siguientes librerías:

- Google api client
- Google HTTP client
- Google OAuth
- Guava
- Jackson core

2.2.1.3.3 OAuth

Oauth es un protocolo de seguridad que permite a los usuarios conceder el acceso de terceros a los recursos de sus páginas web y sus datos sin tener que compartir sus contraseñas. OAuth 1.0 fue publicado en diciembre de 2007 y convertido rápidamente en el estándar de la industria para la delegación de acceso basado en web. Una revisión menor (OAuth 1.0 Revisión A) fue publicada en junio de 2008 para corregir un agujero de seguridad.[17]

Figura 2. 13 Logo OAuth 2.0

2.2.1.3.3.1 Introducción

Muchos coches de lujo cuentan con una llave especial que se da al encargado de aparcamiento y, a diferencia de su llave regular, solo permitirá que el coche pueda ser conducido en un corta distancia mientras que bloquea el acceso al maletero y el teléfono a bordo. A pesar de las restricciones que la llave impone, la idea es muy inteligente. Le da a alguien el acceso limitado a su coche con una llave especial, mientras con la otra llave puede desbloquear todo lo demás.

A medida que la web crece, más y más sitios confían en los servicios distribuidos y de computación en la nube: un laboratorio fotográfico para imprimir tus fotos de Flickr, una red social utilizando la libreta de direcciones de Google para buscar amigos, o una API de una aplicación de terceros para la utilización de servicios multiples.

El problema es que, a fin de que estas aplicaciones puedan acceder a los datos del usuario en otros sitios, piden nombre de usuario y contraseña. Esto no solo requiere la exposición de las contraseñas de usuario a otra persona –a menudo utilizándolas en las mismas contraseñas para banca en línea y otros sitios- sino que también proporcionan esta solicitud de acceso ilimitado a hacer lo que quieran. Pueden hacer cualquier cosa, incluyendo el cambio de las contraseñas de los usuarios y bloquear la cuenta.

OAuth proporciona un método para que los usuarios puedan otorgar a terceros el acceso a sus recursos sin tener que compartir sus contraseñas. También proporciona una manera de dar acceso limitado (en su alcance, duración, etc.).

2.2.1.3.3.2 La experiencia del usuario y otras opciones de emisión de tokens

OAuth consta de dos partes principales: la obtención de una señal pidiendo al usuario que conceda el acceso y el uso de tokens para acceder a sus recursos protegidos. Los métodos para obtener un token de acceso se llaman corrientes. OAuth 1.0 empezó con 3 flujos: aplicaciones basadas en web, clientes de escritorio y dispositivos móviles o limitada. Sin embargo, como la especificación ha ido evolucionado, los tres flujos se fusionaron en uno, que (en papel) permitió a los tres tipos de clientes. En la práctica, el flujo funciona bien para aplicaciones basadas en web, pero ofrece una experiencia inferior en otra parte.

A medida que más sitios comenzaron a usar OAuth, especialmente Twitter, los desarrolladores se dieron cuenta de que el único flujo que ofrece OAuth era muy limitado y, a menudo producen experiencias pobres por parte de los usuarios. Por otro lado, Facebook Connect ofrece un conjunto más rico de flujos, adecuados para las aplicaciones web, dispositivos móviles, y consolas de juegos.

2.2.1.3.3.3 Uso de OAuth 2.0 para aplicaciones instaladas

La versión final de Google OAuth 2.0 es compatible con las aplicaciones que se instalen en un dispositivo (por ejemplo Móvil, Mac, Pc). Estas aplicaciones se distribuyen para las maquinas individuales, y se supone que estas aplicaciones no pueden mantener secretos.

Estas aplicaciones pueden acceder a una API de Google mientras el usuario está presente en la aplicación o cuando se ejecuta la aplicación en modo background durante largos periodos de tiempo sin la interacción directa con el usuario. Este flujo requiere que la aplicación tenga acceso al navegador o la posibilidad de incorporar un control de explorador en la aplicación.

El escenario comienza con la redirección a un navegador (página emergente, o total si es necesario) a una URL de Google con una serie de de parámetros de consulta que indican el tipo de acceso a la API de Google que requiere la aplicación. Al igual que otros escenarios, Google se encarga de la autenticación de usuarios y el consentimiento, el resultado de la secuencia es un código de autorización (en comparación con la entrega directa de un token de acceso).

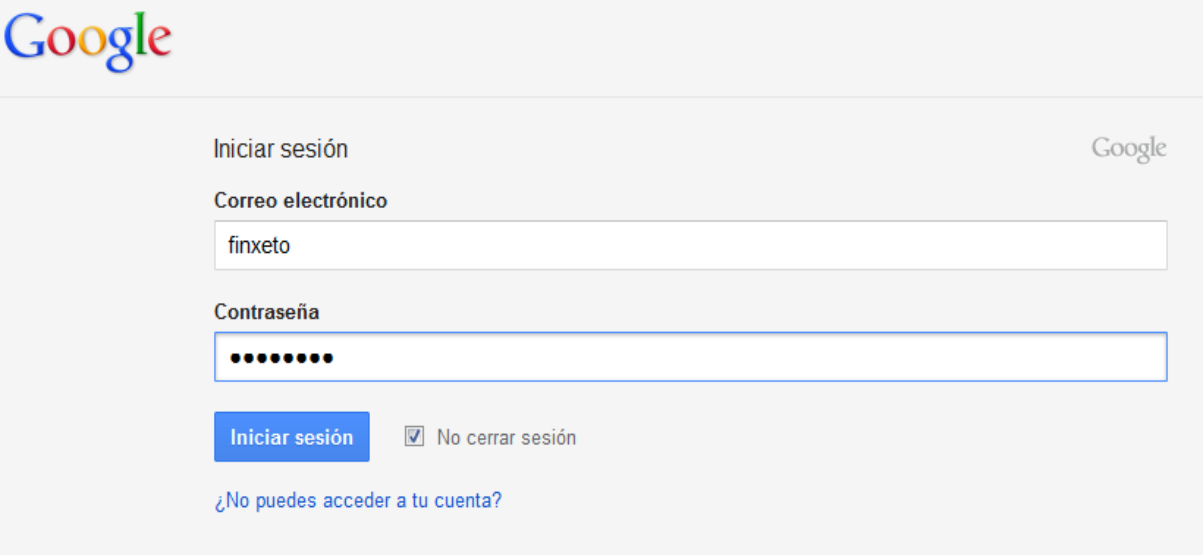


Figura 2. 14 Iniciar Sesión

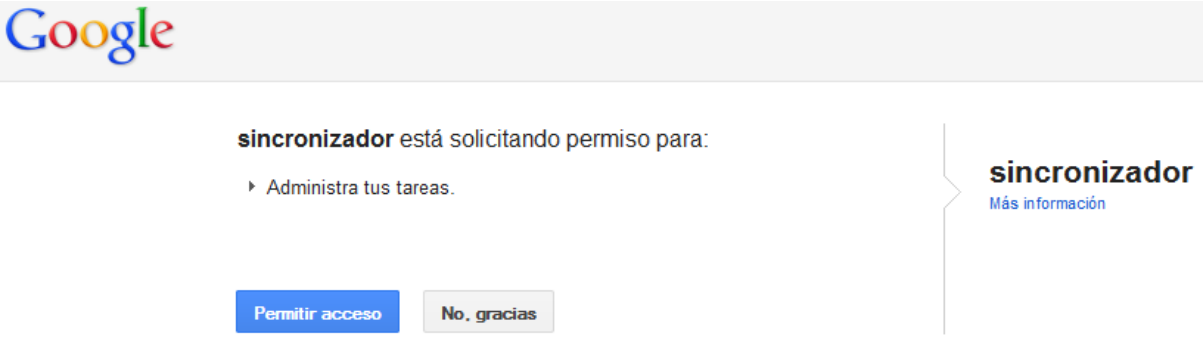


Figura 2. 15 Solicitar Permiso

Google devuelve el código de autorización para la aplicación de servidor web en la cadena de consulta

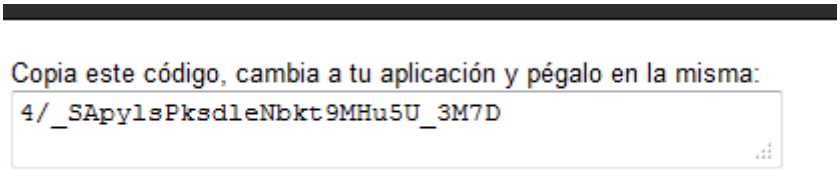


Figura 2. 16 Ejemplo código autenticación

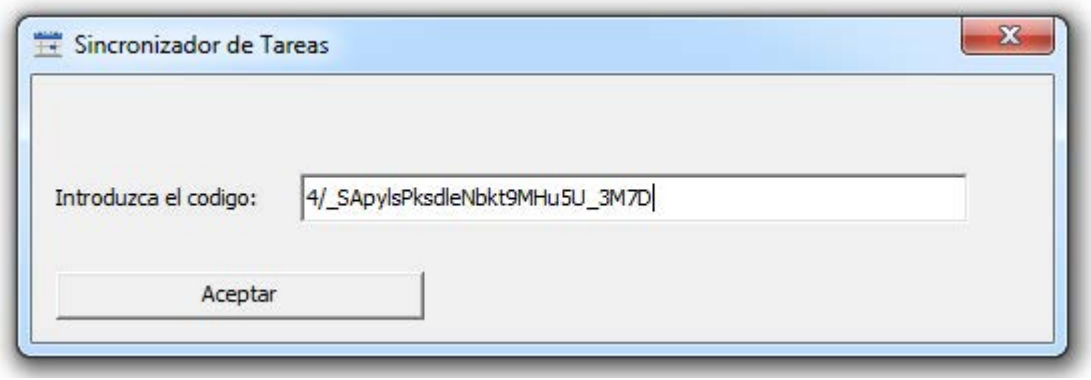


Figura 2. 17 Código introducido

Después de recibir el código de autorización, la aplicación puede intercambiar el código para un token de acceso y, en algunos casos, un token de refresco. La aplicación presenta su client_id y el client_secret (obtenido durante el registro de la aplicación), junto con el código de autorización en la obtención de un token de acceso y el símbolo de renovación.

Campo	Descripción
`code`	El código de autorización devuelto en la solicitud
`client_id`	El id de cliente obtenido durante el registro de la aplicación
`client_secret`	El id de cliente secreto obtenido durante el registro de la aplicación
`redirect_uri`	El URI registrado con la aplicación
`grant_type`	Como está definido en la especificación de OAuth 2.0 este campo debe contener el valor del código de autorización

Tabla 2. 3 Conjunto de datos de autenticación

La solicitud actual se parece a esto:

```
POST /o/oauth2/token HTTP/1.1
Host: accounts.google.com
Content-Type: application/x-www-form-urlencoded

code=4/v6xr77ewYqhvHSyW6UJ1w7jKwAzu&
client_id=8819981768.apps.googleusercontent.com&
client_secret={client_secret}&
redirect_uri=https://oauth2-login-demo.appspot.com/code&
grant_type=authorization_code
```

La respuesta correcta a esta solicitud contiene los siguientes campos:

Campo	Descripción
`access_token`	El token de acceso a la API de Google
`refresh_token`	El token que puede ser usado para obtener un Nuevo token de acceso, y que esta incluido en las aplicaciones de escritorio. Este token es valido hasta que el usuario lo revoca
`expires_in`	El tiempo de vida del token
`token_type`	Indica el tipo de token devuelto. Este campo siempretendrá el valor de *Bearer*

Tabla 2. 4 Datos token

Una respuesta correcta se devuelve en forma de array JSON, similar a la siguiente:

```
{
  "access_token":"1/fFAGRNJru1FTz70BzhT3Zg",
  "expires_in":3920,
  "token_type":"Bearer",
  "refresh_token":"1/xEoDL4iW3cxlI7yDbSRFYNG01kVKM2C-259HOF2aQbI"
}
```

2.2.1.3.4 JSON

JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3rd Edition - Diciembre 1999.

JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

JSON está constituido por dos estructuras:

- ***Una colección de pares de nombre/valor.*** En varios lenguajes esto es conocido como un *objeto*, registro, estructura, diccionario, tabla hash, lista de claves o un array asociativo.

- ***Una lista ordenada de valores.*** En la mayoría de los lenguajes, esto se implementa como arrays, vectores, listas o secuencias.

Estas son estructuras universales; virtualmente todos los lenguajes de programación las soportan de una forma u otra. Es razonable que un formato de intercambio de datos que es independiente del lenguaje de programación se base en estas estructuras.

En JSON, se presentan de estas formas:

Un *objeto* es un conjunto desordenado de pares nombre/valor. Un objeto comienza con `{` (llave de apertura) y termine con `}` (llave de cierre). Cada nombre es seguido por `:` (dos puntos) y los pares nombre/valor están separados por `,` (coma).

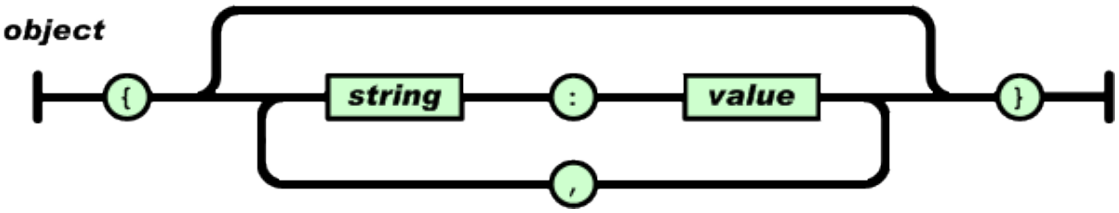


Figura 2. 18 Forma Object

Un *array* es una colección de valores. Un array comienza con `[` (corchete izquierdo) y termina con `]` (corchete derecho). Los valores se separan por `,` (coma).

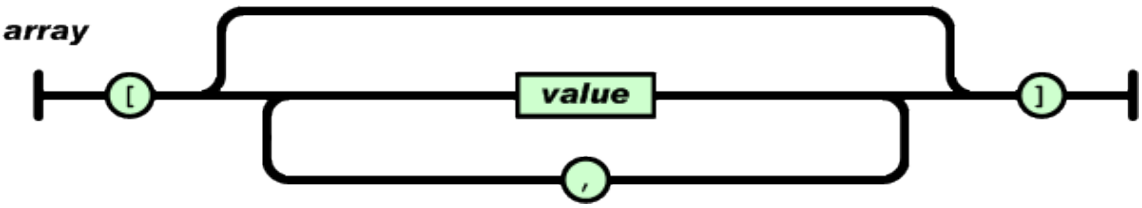


Figura 2. 19 Forma Array

Un *valor* puede ser una *cadena de caracteres* con comillas dobles, o un *número*, o `true` o `false` o `null`, o un *objeto* o un *array*. Estas estructuras pueden anidarse.[18]

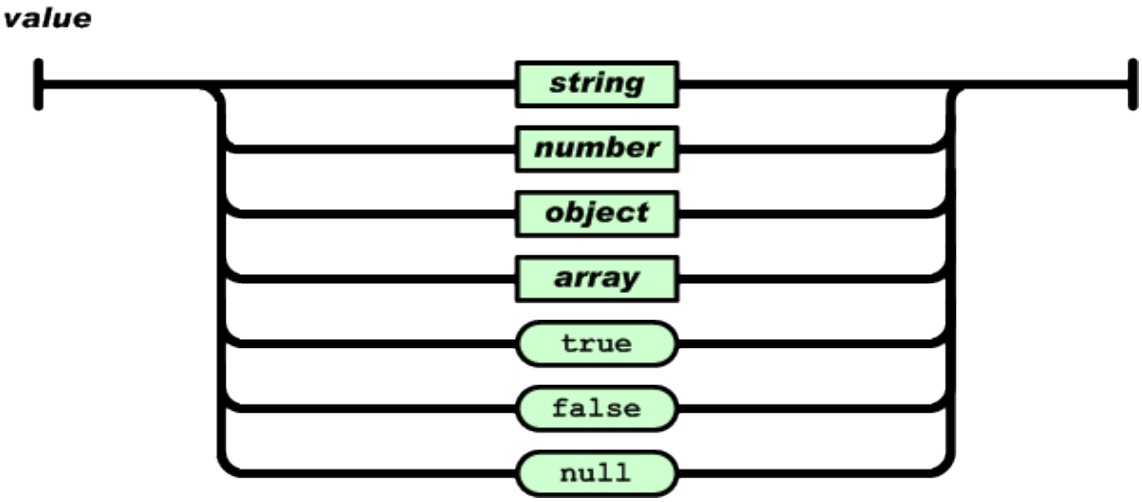


Figura 2. 20 Distintos valores

2.2.1.3.5 JNI

JNI es un mecanismo que nos permite ejecutar código nativo desde java y viceversa.

El código nativo son funciones escritas en un lenguaje de programación como C o C++ para un sistema operativo donde se esta ejecutando la maquina virtual. Llamamos maquina virtual a un entorno de programación formado por:

- El programa que ejecuta los bytecodes Java
- Las librerías (API) de Java

Llamamos host enviorement a tanto al ordenador en que se ejecuta el SO como a las librerías nativas de este ordenador. Estas librerías se suelen escribir en C o C++ y se compilan dando lugar a códigos binarios del hardware donde se ejecuta el host enviorement. De esta forma podemos decir que las maquinas virtuales Java se hacen para un determinado host enviorement.

JNI tiene un interfaz bidireccional que permite a las aplicaciones Java llamar a código nativo y viceversa. La figura que se muestra a continuación muestra gráficamente esta bidireccionalidad.

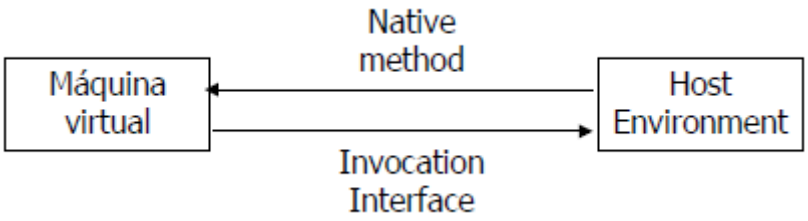


Figura 2. 21 Interacción máq. virtual y host

Es decir JNI soporta dos tipos de interfaces:

1. ***Native methods.*** Que permiten a Java llamar a funciones implementadas en librerías nativas.

2. ***Invocation Interface.*** Que nos permite incrustar una maquina virtual Java en una aplicación nativa. P.e. una maquina virtual Java en un navegador web escrito en C. Para ello, la aplicación nativa llama a librerías nativas de la maquina virtual y luego usa el llamado invocation interface para ejecutar métodos Java en la maquina virtual.

2.2.1.3.5.1 Evolución histórica de JNI

Desde los orígenes de Java se incluía ya una forma de hacer llamadas nativas desde la maquina virtual a las librerías del host enviorement, así como desde el host enviorement a la maquina virtual (en ambas direcciones) pero con 2 problemas:

1. Las llamadas a código nativo desde Java accedían a estructuras C, pero no estaba definida la posición exacta que debían ocupar estos campo en memoria con lo que una llamada nativa en una maquina virtual con las llamadas en otra maquina virtual.

2. Las llamadas nativas al JSDK 1.0 se basaban en el uso de un recolector de basura conservativo, ya que no se controlaba cuando ya no quedaba ningún puntero nativo apuntaba a un objeto Java, con lo que una vez apuntado un objeto Java desde un método nativo éste no se liberaba nunca mas.

Esta forma de llamadas nativas es lo que uso en el JSDK 1.0 para implementar clases que accedían a partes cruciales del host como son java.io.*, java.net.* o java.awt.*

Debido a los problemas arriba mencionados, se decidió implementar una nueva forma de acceso a métodos nativos a la que se llamo JNI, y que es una evolución del JRI (Java Runtime Interface), la técnica de acceso nativo de Netscape diseñada por Warren Harris. Con el JSDK 1.1 surgió la primera versión de JNI, pero todavía las clases Java que accedían al host seguían usando la forma de llamada antigua. En JSDK 1.2 se rescribieron todas las clases que accedían directamente al host usando JNI.

2.2.1.3.5.2 Implicaciones de usar JNI

Una vez que una aplicación usa JNI, se corre el riesgo de perder dos beneficios de la plataforma Java.

En primer lugar, las aplicaciones Java que dependen de JNI ya no pueden funcionar fácilmente con múltiples host enviorements. A pesar de que una parte de la aplicación esté escrita en el lenguaje de programación Java es portable a múltiples host enviorements, será necesario volver a compilar la parte de la aplicación escrita en lenguajes de programación nativos.

En segundo lugar, mientras que el lenguaje de programación Java es de tipo seguro, los lenguajes nativos como C o C++ no lo son. Como resultado, se debe tener especial cuidado cuando se escriben aplicaciones usando JNI. Un mal comportamiento de un método nativo puede corromper toda la aplicación. Por esta razón las aplicaciones Java están sujetas a controles de seguridad antes de aplicar las características de JNI.

Como regla general, se debe construir la aplicación teniendo en cuenta que se deben utilizar el menor número de métodos posibles dentro de las clases de JNI. Esto implica cierto aislamiento entre el código nativo y el resto de la aplicación.

2.2.1.3.5.3 Cuando usar JNI

Antes de embarcarte en un proyecto usando JNI, es valioso tener un par de pasos en los que investigar si existen alternativas que sean más apropiadas. Como se vio en la sección anterior, las aplicaciones que usen JNI tienen una serie de desventajas inherentes a JNI comparadas con aplicaciones escritas estrictamente en el lenguaje de programación Java. Por ejemplo se pierde la seguridad que aporta Java al escribir aplicaciones.

Una serie de enfoques alternativos también permiten que las aplicaciones Java interoperen con código escrito en otros lenguajes. Por ejemplo:

- Una aplicación Java puede comunicarse con una aplicación nativa a través de una conexión TCP / IP o a través de otros mecanismos de comunicación entre procesos (IPC).

- Una aplicación Java puede conectarse a una base de datos heredada a través de la API JDBC.

- Una aplicación Java, podrá hacer uso de las tecnologías de objetos distribuidos, tales como la API de Java IDL.

Una característica común de estas soluciones alternativas es que la aplicación Java y el código nativo pueden residir en diferentes procesos (y en algunos casos en diferentes maquinas). Procesos de separación ofrecen un importante beneficio.

La protección del espacio de direcciones con el apoyo de los procesos permite un alto grado de aislamiento de los fallos. Un fallo en la aplicación nativa no termina inmediatamente con la aplicación Java con la que se comunica a través de TCP / IP.

A veces, sin embargo, puede verse en la necesidad de que una aplicación Java se comunique con un código nativo que reside en el mismo proceso. Este es el momento cuando JNI se vuelve útil. Consideremos, por ejemplo, los siguientes escenarios:

- La API de Java puede no ser compatible con ciertas dependencias del host, características necesarias para la aplicación. Una aplicación que desee realizar, por ejemplo, operaciones especiales de archivo que no son soportadas por la API de Java, sin embargo, es a la vez engorroso e ineficiente el manipular archivos a través de otro proceso.

- Es posible que se desee acceder a una biblioteca nativa existente y no se este dispuesto a pagar por copiar y transmitir datos entre los diferentes procesos. Cargando la aplicación nativa en el mismo proceso es mucho más eficiente.

- Provistos de procesos que abarcan múltiples aplicaciones puede resultar uso de memoria inaceptable. Esto suele suceder si estos procesos deben residir en la maquina del cliente mismo. Cargar una librería nativa existente dentro del proceso que aloja la aplicación requiere menos recursos del sistema que iniciar un nuevo proceso y la carga de la biblioteca en ese proceso.

- Es posible que se desee aplicar una pequeña porción de tiempo crítico de código en un lenguaje de bajo nivel, como por ejemplo ensamblador. Su una aplicación 3D pasa la mayor parte del tiempo en el renderizado de gráficos, puede verse en la necesidad de escribir la parte principal de una biblioteca de gráficos en el código de ensamblador para lograr el máximo rendimiento.

En resumen, se aconseja el uso de JNI, si la aplicación Java debe interoperar con código nativo que reside en el mismo proceso. [19]

2.2.1.3.5.4 Alternativa a JNI para este proyecto

2.2.1.3.5.4.1 Libpst

Como acabamos de ver, en cierta manera se aconseja que no se utilice JNI a no ser que no exista más remedio. Pues bien vamos a estudiar brevemente el remedio para comprobar si se puede utilizar o no.

Esta librería se encuentra de manera Open Source disponible en SourceForge.net, y consiste en una librería para Java que comunica Java con Outlook de manera que puedas leer los emails, contactos, eventos, tareas y demás recursos que pone a tu disposición Outlook. Se valoró esta posibilidad pero tuvo que quedar descartada debido a estas dos condiciones:

- La primera, es que exigía que se importase los recursos que querías leer a un archivo con extensión .PST y se leían los recursos a partir de este archivo. El problema es que hacia molesto el guardar constantemente los recursos en un archivo para poder sincronizar, por lo que en la practica no se sincronizaba, si no que se actualizaba.

- La segunda es que era de solo lectura, por lo tanto, no servía para sincronizar, ya que no dejaba escribir las tareas cogidas del calendario de Google.

Así que no queda más remedio que utilizar JNI para poder sincronizar las tareas, y así conseguir nuestro objetivo. Para ello hemos escogido la librería Jacob que explicaremos a continuación.

2.2.1.3.5.4.2 JACOB (JAva – COm - Bridge)

Jacob es un puente Java-COM que le permite llamar a componentes COM desde Java. Se utiliza JNI para hacer llamadas nativas a las bibliotecas COM y Win 32. El proyecto Jacob comenzó en 199 y está siendo activamente utilizado por miles de desarrolladores en todo el mundo. Como proyecto de código abierto, que se ha beneficiado de la experiencia combinada de estos usuarios, muchos de los cuales han realizado modificaciones en el código y envían de vuelta para su inclusión en el proyecto.

Jacob es una biblioteca Java que permite a las aplicaciones Java comunicarse con Microsoft Windows o con las bibliotecas DLL de Windows. Esto se hace mediante el uso de una DLL personalizada que permite a las clases Java comunicarse con Jacob a través de JNI. La biblioteca y el archivo DLL aíslan al desarrollador de Java de las subyacentes bibliotecas de Windows para que el desarrollador de Java no tenga que escribir código personalizado JNI.

Jacob no se utiliza para crear plugins Active X u otros módulos que viven dentro de las aplicaciones de Microsoft Windows.

2.2.1.3.5.4.2.1 Implementación

La implementación de Jacob de un controlador de automatización COM es muy diferente a la utilizada en la JVM de Microsoft. En Jacob, la clase *com.Jacob.com.Dispatch* es el bloque básico de construcción. Esta clase se utiliza para crear una instancia de un servidor de automatización. La clase *com.Jacob.activeX.ActiveXComponent* simplemente extiende la clase Dispatch para promocionar compatibilidad con la manera en que los servidores de automatización son creados en la JVM de Microsoft. Los métodos de la clase Dispatch son todos estáticos para la compatibilidad con la clase de Microsoft.

El constructor *com.Jacob.Dispatch* toma una cadena como un argumento. Si la cadena contiene dos puntos („:'), entonces trata de interpretarlo como un apodo, de lo contrario, supone que es un ID de Programa. Esto significa que si desea crear un objeto por su CLSID (Estructura del identificados de clase COM) en lugar del ID de programa, entonces se debe utilizar el CLSID. Esto es más limpio que la manipulación de las clases GUID en Java.

La implementación de SafeArray de reproducir la mayoría de la funcionalidad (y algunos de los caprichos) de la implementación de Microsoft. Esto significa, por ejemplo, que tiene el tamaño de la matriz en el constructor antes de llenarlo con los datos, de lo contrario obtendrá un error, y los métodos fromXXXArray no trabajan para matrices multidimensionales (lo cual es consistente con la JVM de Microsoft).

2.2.1.3.5.4.2.2 La DLL de Jacob

Jacob.jar se basa en un archivo DLL que se carga fuera de la ruta de la biblioteca o la ruta de clase. Esto significa que se debe copiar la DLL (existe una DLL para arquitectura de 32 y de 64 bits) adecuada en su ruta de acceso o utilizar las opciones de VM para añadir el directorio de la DLL a la librería de Jacob. Antes de la versión 1.14M6, el nombre de la DLL Jacob siempre fue "Jacob.dll".

Esto hizo difícil verificar que Jacob cargaba la DLL correcta cuando existían multiples copias de Jacob instaladas en un solo sistema. También se presta a confusión en sistemas de 64 bits donde el bit 32 y el 64 tienen la misma forma. A partir de la versión 1.14M6, la biblioteca de Jacob incluye un gestor que selección un archivo DLL con el nombre apropiado basado en la liberación de Jacob y la plataforma.

La convención de nomenclatura para la DLL de Jacob es la siguiente:

Jacob <plataforma><versión>.dll

El código esta escrito para que la librería DLL se cargue solo una vez. Esto funciona bien en una aplicación estándar pero puede causar serios problemas si se carga desde mas de una ruta de clase.

2.2.1.3.5.4.2.3 Ciclo de vida de un objeto Jacob

La versión de Jacob 1.7 (que es la que utiliza el gadget) implementa un nuevo modelo de hilos que es más compatible con los objetos COM. Existe también una incompatibilidad entre el ciclo de vida de un objeto Java y los objetos COM. Los objetos COM viven y mueren por su cuenta de referencia, mientras que los objetos Java son recogidos por el colector de basura, basado en algoritmos que se ocultan para el usuario.

Antes de la versión 1.6, los objetos Jacob que envuelven a los objetos COM tenían el método *finalize()* que llamaban a un método nativo para la liberación del objeto COM. Esto tenía muchos problemas ya que el colector de basura puede tardar mucho tiempo en arrancar y el consumo de recursos crecía exponencialmente.

En la versión 1.7 todos los objetos Jacob que envuelven los objetos COM se extienden a la clase *com.jacob.com.JacobObject*. Este objeto tiene un código especial para registrarse a si mismo con la clase *com.jacob.com.ROT* que representa una Tabla de Objetos en Ejecución (ROT Running Object Table). En esta tabla se asigna un mensaje privado al conjunto de JacobObjects creados en ese hilo.

Por lo tanto cuando se llama a *ComThread.Release ()*, la tabla ROT comprueba si ese hilo ha creado algún objeto, y esos objetos son liberados llamando al método nativo de liberar (que es publico).

Este método de administración de la duración del ciclo de vida une al ciclo de vida del hilo en lugar de al colector de basuras. El *JacobObject* todavía tiene finalizadores, pero no realizan la llamada al método nativo de finalización. [20]

2.2.1.3.6 Base de Datos

En realidad no se utiliza ningún gestor de base de datos, ni SQL ni nada por el estilo, ya que debido a la simplicidad del uso que se le da no es necesario. La base de datos, consiste en dos archivos con extensión .TXT en el que se guardan las tareas que se encuentran en cada sincronización en ambos calendarios. Esto se hace por que si no, no se tendría una consistencia sobre que tareas se han creado y borrado. Es decir, si no se lo que había antes no podre adivinar que es lo que ha cambiado. El formato que tienen ambos archivo es el siguiente:

<Titulo Tarea> : El titulo de la tarea que se esta tratando, se utiliza como identificador único.

<Formato Fecha> : El formato de cada calendario es diferente, y en este archivo se ha mantenido igual ya que así podremos diferenciar de donde viene esta tareas, es decir donde se ha creado. Identificador secundario.

<Descripción> : La descripción de la tarea.

<Realizada> : Si ya esta realizada o no.

Ejemplo:

Comprar leche
2012-02-29T00:00:00.000Z
Andrés acuérdate de comprar leche.
No

De esta forma podemos mantener la consistencia entre ambos calendario, ya que si por ejemplo tenemos una tarea creada en Outlook y que esta sincronizada en Google, si esta tarea es borrada en Google, lógicamente, la tendremos que borrar también del calendario de Outlook, y se logra gracias a esta pequeña base de datos, si es que se puede llamar así, ya que si al leer las tareas de Google no leemos esta tarea y si esta en el archivo de Google significa que a sido borrada, por lo tanto debemos borrarla del calendario de Outlook.

2.2.1.3.7 Swing

2.2.1.3.1 Introducción

Para poder apreciar la importancia de Swing, haremos primero una introducción a las interfaces de usuario y al AWT.

El interfaz de usuario es la parte del programa que permite a éste interactuar con el usuario. Las interfaces de usuario pueden adoptar muchas formas, que van desde la simple línea de comandos hasta las interfaces gráficas que proporcionan las aplicaciones más modernas. El interfaz de usuario es el aspecto más importante de cualquier aplicación. Una aplicación sin un interfaz fácil, impide que los usuarios saquen el máximo rendimiento del programa. Java proporciona los elementos básicos para construir interfaces de usuario a través del AWT, y opciones para mejorarlas mediante Swing, que sí permite la creación de interfaces de usuario de gran impacto y sin demasiados quebraderos de cabeza por parte del programador.

Al nivel más bajo, el sistema operativo transmite información desde el ratón y el teclado como dispositivos de entrada al programa. El AWT fue diseñado pensando en que el programador no tuviese que preocuparse de detalles como controlar el movimiento del ratón o leer el teclado, ni tampoco atender a detalles como la escritura en pantalla. El AWT constituye una librería de clases orientada a objeto para cubrir estos recursos y servicios de bajo nivel. Debido a que el lenguaje de programación Java es independiente de la plataforma en que se ejecuten sus aplicaciones, el AWT también es independiente de la plataforma en que se ejecute.

El AWT proporciona un conjunto de herramientas para la construcción de interfaces gráficas que tienen una apariencia y se comportan de forma semejante en todas las plataformas en que se ejecute. Los elementos de interfaz proporcionados por el AWT están implementados utilizando toolkits nativos de las plataformas, preservando una apariencia semejante a todas las aplicaciones que se creen para esa plataforma. Este es un punto fuerte del AWT, pero también tiene la desventaja de que un interfaz gráfico diseñado para una plataforma, puede no visualizarse correctamente en otra diferente. Estas carencias del AWT son subsanadas en parte por Swing, y en general por las JFC.

2.2.1.3.2 Swing, el AWT y las JFC

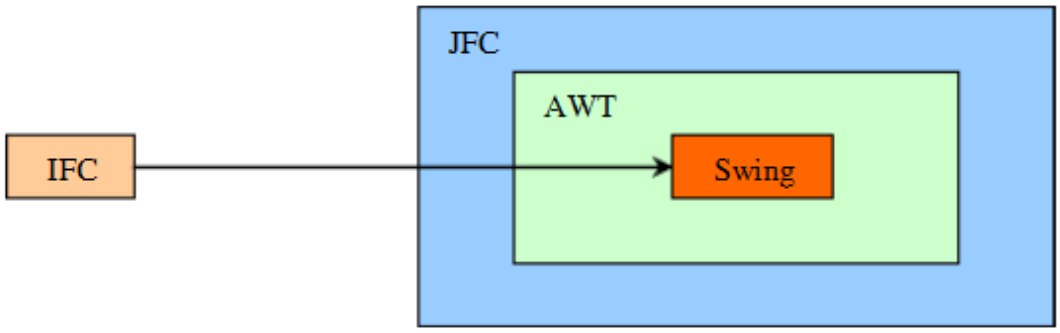


Figura 2. 22 Relación entre Swing, AWT y JFC

La Figura 2.22 muestra la relación existente entre Swing, el AWT y las JFC. Las JFC subsumen y amplían el AWT original, y constan de las siguientes API principales:-

- AWT.-
- Swing.-
- Java 2D.-
- Drag-and-Drop.-
- Accessibility.

Aunque Swing esté separado del AWT, se implementa en términos de clases AWT básicas. El AWT proporciona la interfaz entre el sistema de ventanas nativo subyacente y los componentes GUI de Java. Swing utiliza esta interfaz, pero no se apoya en componentes del AWT para hacer uso de objetos nativos. En lugar de ello, los componentes Swing están escritos en Java puro. Esto ofrece ventajas significativas. Permite a los componentes Swing ser independientes del sistema de ventanas nativo, lo cual implica que pueden ejecutarse en cualquier sistema de ventanas que admita el AWT. También permite a los componentes Swing ser independientes de cualquier limitación de los sistemas de ventanas nativos. Esta independencia permite a Swing controlar y adaptar su aspecto y sensación (de ahí la aparición de Plastic Looks & Feels).Entre los componentes nuevos que incluye Swing hay desde paneles tabulados y bordes estilizados hasta barras deslizadoras y efectos giratorios. Estos componentes nuevos, en sí mismos, hacen que Swing constituya un agregado de primera magnitud a la API Java.

Swing también ofrece una implementación de Java puro de muchos de los componentes tradicionales del AWT. Estos componentes tienen la misma funcionalidad que los componentes del AWT y todas las ventajas de Swing. Swing es compatible con el AWT, y los componentes Swing se pueden utilizar con los componentes del AWT. Sin embargo, los componentes Swing sólo se pueden usar con el modelo de eventos del JDK 1.1. No admiten el modelo de eventos del JDK 1.0.La arquitectura PL&F de Swing facilita la personalización de tanto del aspecto como del comportamiento de cualquier control Swing o de cualquier

grupo de estos controles. Swing también incorpora varios L&F predefinidos, entre los que re incluye el Metal L&F predeterminado, el Motif L&F y el Windows L&F.

El problema que surge con Swing en Eclipse, es que a diferencia de Netbeans, no dispone de un editor de fácil uso, con el que poder crear la interfaz gráfica, por lo tanto vamos a tener que echar mano de algún plugin que se incluya dentro de Eclipse para poder crear la interfaz. Para ello hemos elegido Jigloo que es gratuito siempre y cuando no se utilice para un uso comercial. Vamos a describirlo a continuación. [21]

2.2.1.3.8 Jigloo y Swing

Jigloo Cloud Garden de GUI Builder es un plugin para el IDE de Eclipse Java y WebSphere Studio, que le permite crear y gestionar tanto las clases de Swing y SWT GUI. Jigloo crea y administra el código para todas las partes de Swing o SWT interfaces gráficas de usuario, así como el código para manejar eventos, y le muestra la GUI, como se esté construyendo. Jigloo analiza los archivos de clase Java para la construcción de la forma que utiliza el diseño de su interfaz gráfica de usuario (de ida y vuelta), para que pueda trabajar en las clases que se generaron por otros constructores de GUI o entornos de desarrollo, o las clases de codificación manual. También puede convertir a partir de una interfaz gráfica de usuario GUI de Swing a un SWT y viceversa.

Jigloo es sencillo, rápido, potente, fácil de usar y totalmente integrado con Eclipse. Esto puede llevar a importantes ahorros de tiempo para el desarrollo de interfaz gráfica de usuario y las tareas de mantenimiento.

Jigloo es altamente personalizable: las partes de su código que Jigloo analizará puede ser restringido, y las clases de las que se crean instancias cuando Jigloo analiza el código y construye el editor de formularios se puede especificar que utilice el uso de patrones. El código generado por Jigloo también se pueden personalizar, y el código existente puede ser re-organizado para seguir el estilo preferido (por ejemplo, utilizando captadores de elementos de la GUI, o separar los elementos de las líneas en blanco, aparatos ortopédicos o comentarios etiquetados).

Las clases personalizadas se pueden agregar a las formas, y JavaBeans con personalizadores y propiedades personalizadas son compatibles. Además, Jigloo soporta la herencia visual - se puede diseñar clases que se extienden a otras clases personalizadas, que pueden ser públicas, abstracta o no pública. La navegación entre el código y los editores de la forma es muy fácil - con Jigloo destacando la sección correspondiente del código cuando el editor de formularios tiene el foco, o el elemento de forma relevante cuando el editor de código tiene el foco.

No es necesario un conocimiento básico del Swing y de los componentes SWT, pero no es esencial, se puede acceder fácilmente al javadoc (mediante una opción de botón derecho del ratón) directamente desde el editor de interfaz gráfica de usuario. [22]

2.3 Estudio UML

2.3.1 Restricciones Obligatorias

2.3.1.1 Restricciones de Solución

	Descripción	***Análisis razonado***	***Criterios de aceptación***
SOL 1	El producto hará uso de la tecnología para la que se quiere solucionar el problema, es decir, hará uso de Microsoft Outlook y de Google Calendar.	Lógicamente se quiere solventar el problema de sincronización de tareas para estos dos programas, por lo tanto tendrá que interactuar con ellos.	La sincronización se realiza para estas dos tecnologías en concreto, no para ninguna otra.
SOL 2	La licencia usada para Microsoft Outlook es original.	Bajo ningún concepto, este programa puede ser utilizado con números de serie de Microsoft Outlook que no hayan sido adquiridos de manera legítima.	La clave del producto debe de ser original, de ninguna de las maneras queremos dar a entender que apoyamos la piratería.
SOL 3	No es necesario adquirir ninguna licencia aparte de la que ya se tenga.	Para poder usar este programa, el cliente no tendrá necesidad alguna de adquirir una nueva clave para poder acceder a sus datos personales desde el programa en cuestión.	Siempre y cuando no se tenga una licencia no valida, no será necesario hacer al cliente pagar por una nueva.
SOL 4	El entorno de programación de la aplicación ha de ser de libre uso, es decir, no se deberá adquirir ningún programa cuya licencia tenga un coste para el cliente.	Al tratarse de un gadget, un programa en términos generales pequeño, no conviene comprar ninguna licencia para poder desarrollarlo.	El producto ha de ser desarrollado dentro de la gama de entornos de programación de libre uso.
SOL 5	El programa ha de ser liviano.	El programa pretende ser una ayuda para facilitar la labor de los trabajadores, no una carga pesada que ralentice el ordenador, hasta tal punto que desespere al trabajador por usarlo.	El programa no pesara más de 10 megas, y deberá realizar sus tareas en un tiempo menor a 1 minuto.
SOL 6	El programa no debe molestar con mensajes innecesarios, debe ser los mas silencioso y oculto posible.	El programa no debe de tener una ventana molestando en medio del escritorio, si no que debe de pasar desapercibido.	El programa se minimizara en la barra del sistema de Windows.
SOL 7	El producto funcionará solo con entornos Windows.	Hoy en día la gran mayoría de empresas utilizan Windows, así que nos centraremos en este sistema operativo.	Sera aceptable cualquier sistema operativo de Windows partiendo del XP en adelante (Sin incluir Windows 8).

	Descripción	***Análisis razonado***	***Criterios de aceptación***
SOL 9	El producto funcionará con Microsoft Outlook versión 2007 y 2010.	Las versiones anteriores pueden plantear problemas con este producto debido a los cambio de estas dos versiones respecto de sus anteriores versiones.	Serán aceptables estas dos versiones de Microsoft Outlook.
SOL 10	Debido al uso de protocolos de Google en cuanto a autenticación, el programa deberá sincronizar, como mínimo en un tiempo de 30 minutos.	Si no se cumple esta restricción, probablemente ocurran errores, que no tienen que ver con la implementación de este producto, si no con la autenticación por parte de Google, y no se asegura su correcto funcionamiento.	El programa sincronizara automáticamente, como mínimo, cada 30 minutos.

Tabla 2. 5 Restricciones de Solución

2.3.1.2 Aplicaciones de colaboración o sociedad

2.3.1.2.1 Microsoft Outlook

Descripción:

Microsoft Outlook ofrece herramientas de primera calidad para la administración del correo electrónico personal y de negocios a más de 500 millones de usuarios en todo el mundo. Un perfil de correo electrónico consta de cuentas de correo electrónico, archivos de datos e información sobre dónde se almacena el correo electrónico.

Los perfiles de correo electrónico son lo que usa Outlook para recordar qué cuentas de correo electrónico utiliza y donde se guardan los datos correspondientes a cada cuenta. Cada perfil proporciona a Outlook la siguiente información:

- **Qué información de cuenta se debe usar** Esta información incluye el nombre de usuario, un nombre para mostrar, el nombre del servidor de correo electrónico y la contraseña de la cuenta del proveedor de servicios de Internet (ISP).

- **Dónde se entregan y se almacenan los datos de correo electrónico** En Outlook, los datos se entregan y se almacenan en el servidor de correo electrónico o en un archivo .pst en el equipo. Estos datos incluyen reglas, mensajes, contactos, calendarios, notas, tareas, diarios, carpetas de búsqueda y otros ajustes.

Los perfiles de correo electrónico de Outlook se almacenan en el Registro de Windows. Al iniciarse Outlook, recupera la información del perfil en el Registro.

Este producto solo interactuará con las tareas. Vamos a ver que campos contienen las tareas:

Asunto: Permite ingresar una descripción de la tarea.

Vencimiento: Se debe elegir entre el botón ninguno (cuando no existe vencimiento para la tarea) o vencimiento y comienzo donde se eligen las fechas correspondientes.

Estado: Se indica si la tarea esta en curso, comenzada, etc.

Prioridad: Este ítem permite establecer el nivel de prioridad de la tarea (alta, baja o normal).

Aviso: Si se desea poner un aviso sobre la tarea, es posible asociarle sonidos.

Propietario: Especifica quien es el autor de la tarea.

Categoría: Se utiliza este ítem para poder agrupar tareas según algún criterio.

2.3.1.2.2 Google Calendar

Descripción:

Google Calendar es un calendario online a través del cual puedes crear eventos y tareas para organizar tu vida laboral o familiar, y tener todos los datos disponibles en internet, para poder consultarlos desde cualquier lugar.

Este calendario se asocia a una cuenta de correo Gmail, es completamente gratuita, lo único que necesitas es una cuenta de correo.

Este producto, al igual que el anterior, solo interactuará con la parte relacionada con las tareas, cuyos campos son:

Nombre: Id para identificar cada tarea.

Vencimiento: Fecha en la que termina la tarea.

Descripción: Contenido de la tarea.

Estado: Completada o no completada.

Debido a que las tareas de Google Calendar poseen menor información que las tareas de Outlook, nuestro producto solo sincronizará los campos de Google Calendar, ya que no existe la posibilidad de introducir en Google Calendar todos los campos que posee el Outlook, por lo tanto, nos debemos ceñir a los campos que sean idénticos de un programa a otro para poder realizar la sincronización, que en este caso son los 4 campos citados anteriormente.

2.3.2 Requisitos Funcionales

	Descripción	***Razón fundamental***	***Criterio de aceptación***	***Satisfacción cliente***	***Insatisfacción cliente***	***Prioridad***	***Materiales de soporte***
FUN 1	El programa deberá sincronizar las tareas de Google Calendar a Microsoft Outlook, de Microsoft Outlook a Google Calendar, y en ambos sentidos a la vez.	Poder pasar las tareas de la lista de tareas de un calendario a otro con todas sus posibilidades de sincronización.	Las tareas que se encontraban en la lista de tareas de un calendario se reflejen en la lista de tareas del otro calendario.	10	0	1	JACOB, JNI, Google Task API
FUN 2	El programa debe poder acceder a los datos contenidos en la cuenta de correo de Gmail del usuario.	Para poder sincronizar las tareas, primero debemos poder acceder a ellas para poder decidir cuales han de ser sincronizadas.	Poder coger las tareas que se encuentran almacenadas en la cuenta de Gmail y poder almacenarlas en nuestro producto para poder analizarlas.	10	0	1	Google Task API
FUN 3	El programa debe lanzar el autenticador de Gmail para que el usuario pueda autenticarse la primera vez que se mete en su cuenta de correo.	Para poder coger las tareas de la cuenta de Gmail, el programa deberá lanzar el autenticador de Gmail para poder autenticar al usuario en su cuenta de correo.	Poder coger las tareas que se encuentran almacenadas en la cuenta de Gmail y poder almacenarlas en nuestro producto para poder analizarlas.	6	4	1	OAuth 2
FUN 4	El programa deberá guardar el token de autorización que Google entrega al autenticarse en la cuenta de correo Gmail.	El token es fundamental, para poder ir accediendo a los datos sin necesidad de estar autenticando constantemente al usuario, por ello, cuando nos hacen entrega del token, deberemos guardarlo para su uso en posteriores sincronizaciones	Almacenar el token, para su posterior uso.	8	4	1	OAuth 2
FUN 5	El producto ha de ser capaz de crear tareas, modificar o eliminar tareas dentro de la lista de tareas de Google Calendar, según el usuario las haya creado, modificado o borrado.	El usuario crea, modifica o borra tareas de la lista de Outlook, y el programa deberá efectuar estos cambios en la lista de tareas de Google Calendar.	Creación, modificación o borrado de tareas en la lista de tareas de Google Calendar.	10	0	2	N/A

	Descripción	***Razón fundamental***	***Criterio de aceptación***	***Satisfacción del cliente***	***Insatisfacción del cliente***	***Prioridad***	***Materiales de soporte***
FUN 6	El producto ha de ser capaz de acceder a los datos contenidos en la lista de tareas de Google Calendar sin necesidad de acceder a la cuenta a través de una autenticación, con el token que se guardó anteriormente.	Cuando anteriormente guardamos el token de acceso, solo se necesita un acceso para conseguir el token, el programa tendrá acceso ilimitado a los datos sin necesidad de autenticar constantemente al usuario y sin necesidad de abrir el navegador web para ello.	Acceso sin autenticación, una vez autenticado por primera vez, y sin necesidad de que se abra el navegador web.	7	4	3	N/A
FUN 7	El programa debe poder acceder a los datos contenidos en el archivo .pst de Microsoft Outlook.	Para poder sincronizar las tareas, primero debemos poder acceder a ellas para poder decidir cuales han de ser sincronizadas.	Poder coger las tareas que se encuentran almacenadas en el archivo con extensión **.pst** de Microsoft Outlook y poder almacenarlas en nuestro producto para poder analizarlas.	7	5	1	JACOB, JNI
FUN 8	El producto ha de ser capaz de crear tareas, modificar o eliminar tareas dentro de la lista de tareas de Microsoft Outlook, según el usuario las haya creado, modificado o borrado.	Para poder sincronizar las tareas, primero debemos poder acceder a ellas para poder decidir cuales han de ser sincronizadas.	Creación, modificación o borrado de tareas en la lista de tareas de Microsoft Outlook.	10	0	1	N/A
FUN 9	El programa tiene que permitir la sincronización de tareas de manera automática, es decir, debe tener un contador de tiempo, y cuando se alcance el tiempo establecido por el usuario, se tendrá que sincronizar automáticamente.	Para no entrar en conflicto con un requisito obligatorio, el programa tiene que sincronizar cada X minutos elegidos por el usuario, siempre y cuando el tiempo sea mayor o igual a 30 minutos.	Sincronización automática cada X minutos.	6	6	2	N/A

	Descripción	***Razón fundamental***	***Criterio de aceptación***	***Satisfacción del cliente***	***Insatisfacción del cliente***	***Prioridad***	***Materiales de soporte***
FUN 10	El programa tiene que permitir la sincronización de tareas de manera manual.	Si el programa no se esta ejecutando constantemente, se debe permitir sincronizar cuando el usuario lo crea conveniente.	Sincronización manual.	6	6	2	N/A
FUN 11	La aplicación se debe esconder en la barra de sistema cuando el usuario la minimice.	Para no estar en primera línea del escritorio, al minimizar la aplicación, se deberá posicionar en la barra del sistema.	Minimización en la barra de sistema.	10	5	3	N/A

Tabla 2. 6 Requisitos Funcionales

2.3.3 Requisitos De Usabilidad

	Descripción	***Razón fundamental***	***Criterio de aceptación***	***Satisfacción del cliente***	***Insatisfacción del cliente***	***Prioridad***	***Materiales de soporte***
USA 1	La aplicación debe tener un estilo sobrio, comercial y empresarial, sin distracciones.	Debe mantener una estética parecida a un programa de Windows, sin colores exagerados, ni posibles distracciones, como anuncios o links innecesarios.	Interfaz grafica sobria, y con cierto estilo a los programas de Windows.	8	8	4	N/A
USA 2	El programa debe ser lo más eficaz posible para poder realizar su cometido, sin excesivos pasos para poder lograr la sincronización, como máximo 5 clicks en el caso de que tengas que autenticarte o 3 si ya estas autenticado.	Cuán rápidamente puede el usuario usar el producto con exactitud.	No superar el número máximo de clicks.	10	2	1	N/A
USA 3	El producto debe ser lo más simple posible	Lograr que el usuario lo utilice a menudo.	Simplicidad	10	4	1	N/A

	Descripción	***Razón fundamental***	***Criterio de aceptación***	***Satisfacción del cliente***	***Insatisfacción del cliente***	***Prioridad***	***Materiales de soporte***
USA 4	El producto es veloz a la hora de realizar sus tareas (no más de 3 min en sincronizar)	El producto tiene que ser rápido al sincronizar las tareas	Tiempo inferior a 1 minuto.	10	5	1	N/A
USA 5	El programa no debe ocupar mucha memoria (el programa debe pesar en total menos de 5MB).	El programa ha de ser liviano para no ocupar mucho espacio en disco.	Peso inferior a 5Mbs	10	0	1	N/A
USA 6	El producto cumple la sincronización sin fallos y en caso de algún fallo, mostrar un mensaje que muestre cual es el fallo que se ha producido.	No se produzcan fallos mientras sincroniza y en caso de que lo haga que muestre un mensaje con el fallo producido.	Sincronización sin fallos.	8	6	1	N/A

Tabla 2. 7 Requisitos De Usabilidad

2.3.4 Requisitos No Funcionales

	Descripción	*Razón fundamental*	*Criterio de aceptación*	*Satisfacción del cliente*	*Insatisfacción del cliente*	*Prioridad*	*Materiales de soporte*
NOFUN 1	Los datos han de ser guardados en una base de datos para poder mantener una consistencia para saber que cambios han sido realizados	Poder saber en cada sincronización que cambios se han producido para saber como proceder a la sincronización.	Consistencia en la sincronización	10	3	1	BBDD (en este caso un archivo de texto es más que suficiente)
NOFUN 2	Es necesaria una conexión a internet para poder realizar la sincronización con Google Calendar.	Sin conexión a internet no se puede realizar la sincronización	Conexión a internet	10	0	1	N/A
NOFUN 3	En caso de instalar el programa en Archivos de Programa, en la versión de Windows 7 se necesitan establecer permisos de Root para poder realizar la sincronización.	Permisos de Root para poder realizar la sincronización.	Se establecen permisos de Root.	8	6	3	N/A
NOFUN 4	El sincronizador funcionará con cualquier navegador instalado en el ordenador, ya que reconoce cual usa para poder lanzar la autenticación	Lanzar el navegador para poder realizar la autenticación	Lanza el navegador predeterminado del usuario.	9	2	3	BrowserLauncher

Tabla 2. 8 Requisitos No Funcionales

2.3.5 Diagrama de Arquitectura

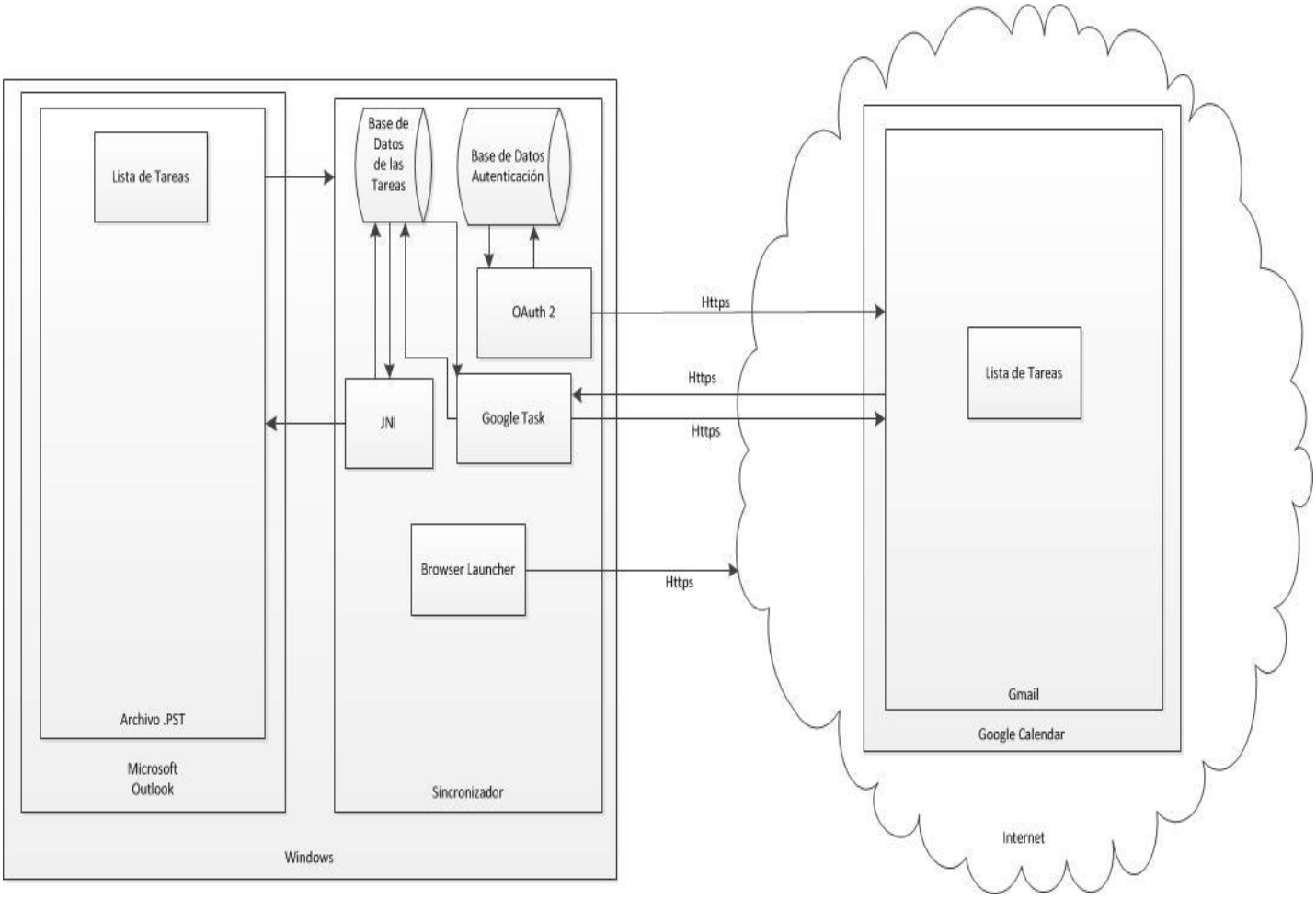


Figura 2. 23 Diagrama de Arquitectura

2.3.5 Diagrama de Clases

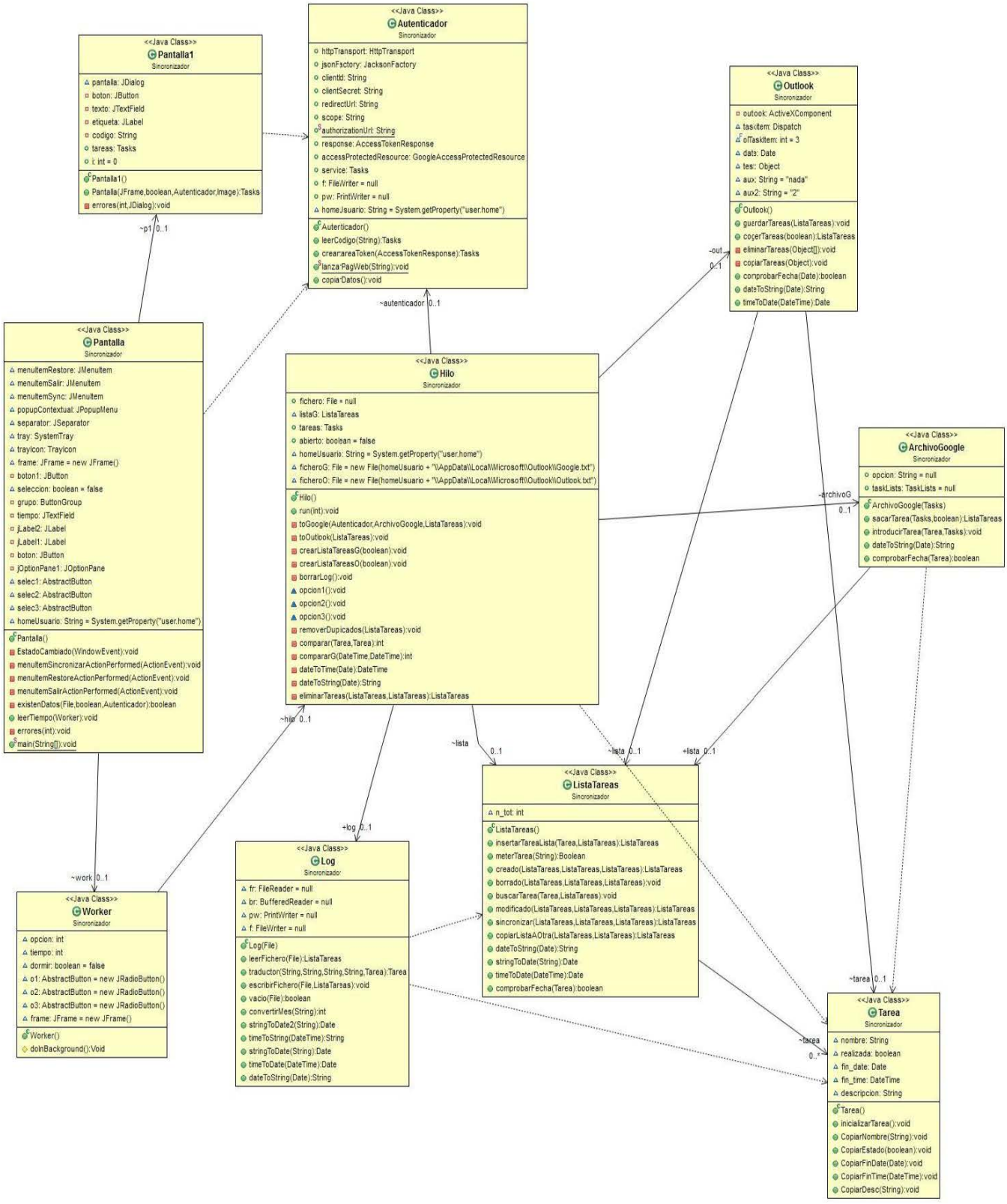
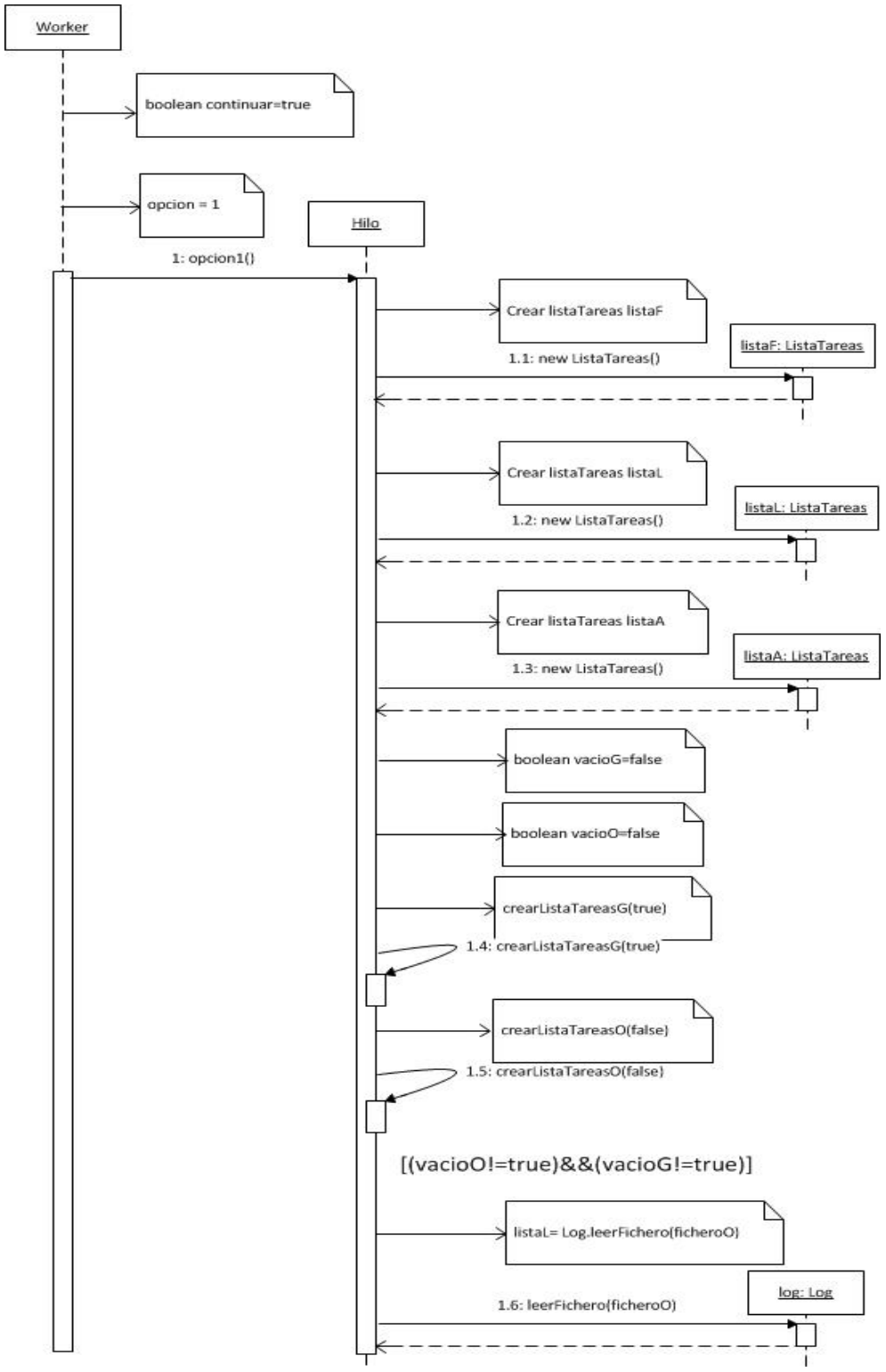
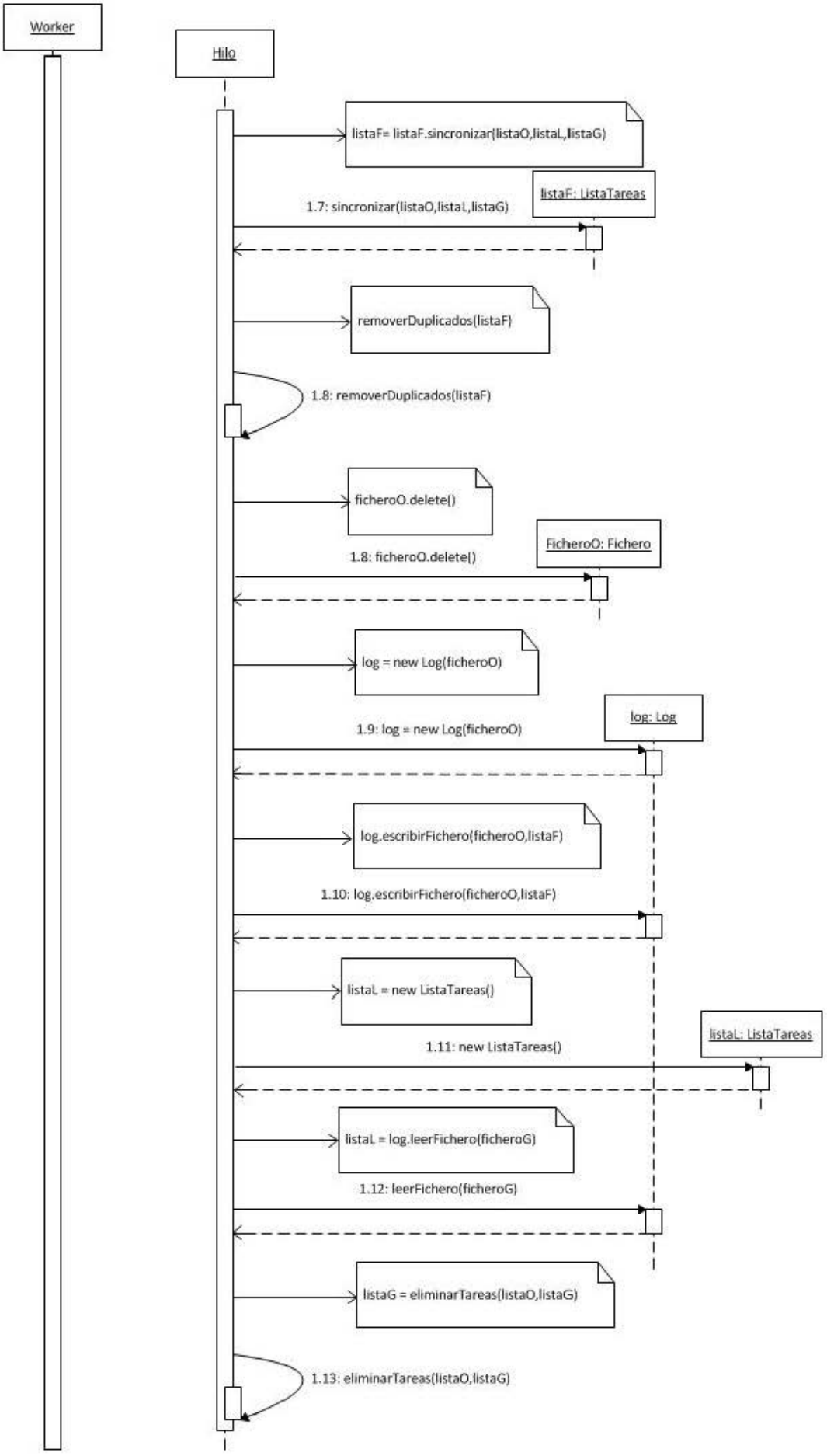


Figura 2. 24 Diagrama de Clases

2.3.5 Diagrama de Secuencia




Worker
Hilo
listaF= listaF.sincronizar(listaO,listaL,listaG)
listaF: ListaTareas
1.7: sincronizar(listaO,listaL,listaG)
removerDuplicados(listaF)
1.8: removerDuplicados(listaF)
ficheroO.delete()
FicheroO: Fichero
1.8: ficheroO.delete()
log = new Log(ficheroO)
log: Log
1.9: log = new Log(ficheroO)
log.escribirFichero(ficheroO,listaF)
1.10: log.escribirFichero(ficheroO,listaF)
listaL = new ListaTareas()
listaL: ListaTareas
1.11: new ListaTareas()
listaL = log.leerFichero(ficheroG)
1.12: leerFichero(ficheroG)
listaG = eliminarTareas(listaO,listaG)
1.13: eliminarTareas(listaO,listaG)

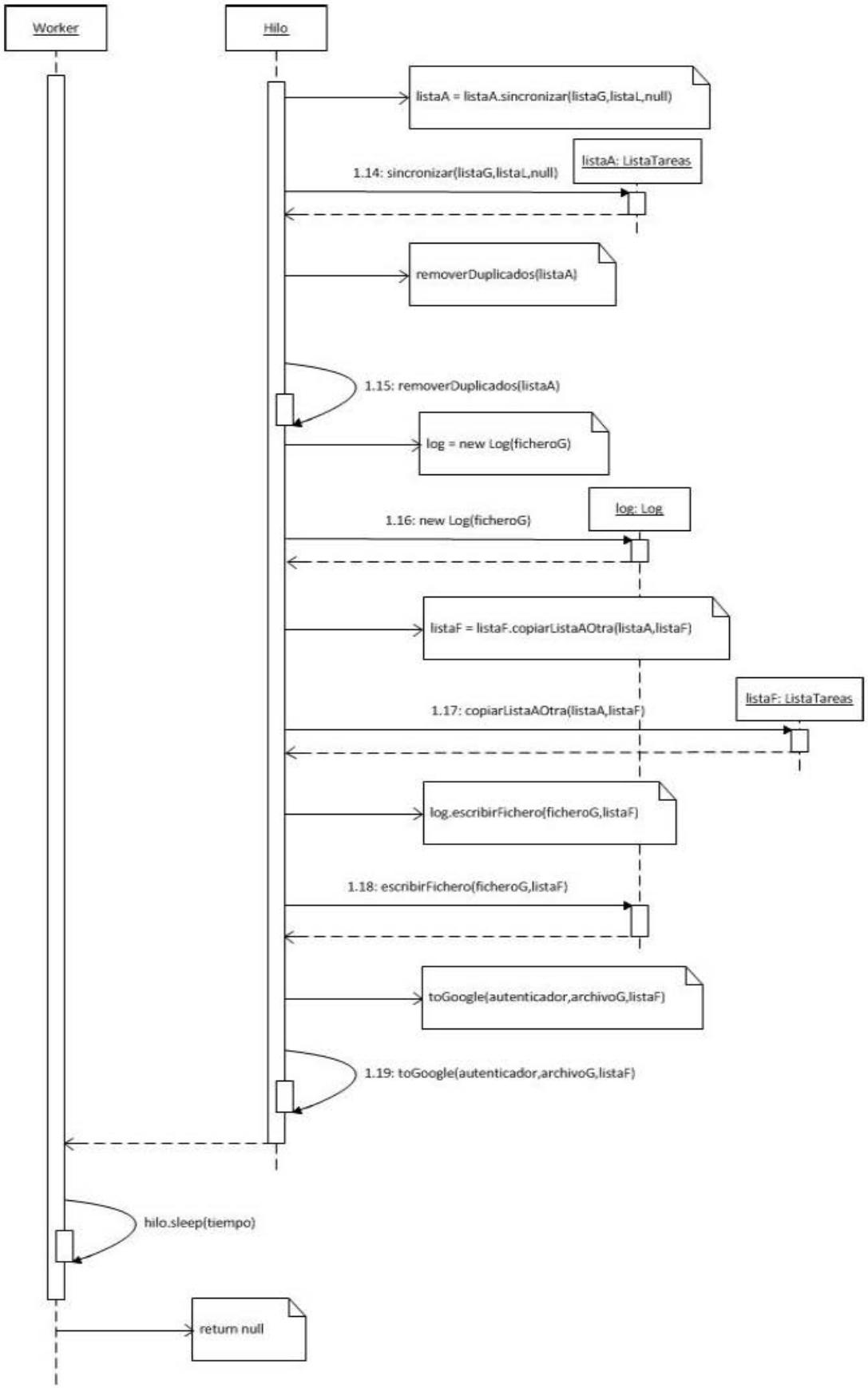


Figura 2. 25 Diagrama de Secuencia

2.4 Visión de alto nivel

La aplicación desarrollada en el proyecto está pensada para obtener información desde cualquier Outlook instalado en cualquier ordenador, tanto si esta protegido con clave como si no lo está, tanto si esta abierto el Outlook como si esta cerrado. Y lo mismo para Google, es decir, con que te hayas autenticado desde el gadget, se obtiene el token necesario para poder acceder a los datos contenidos con Google, y ya puedes sincronizar tus tareas desde este pequeño gadget.

2.3.1 Medios Software para realizar el proyecto

- La plataforma en la que se ha desarrollado el proyecto es **Eclipse Indigo versión 3.7.1**
    - El proyecto se ha realizado con el lenguaje de programación **Java, versión 6.**
    - Para la construcción del GUI se ha utilizado **Jigloo versión 4.6.4.**
    - La librería de Google usada para la sincronización con este es **Google Client Api para Java versión 1.6.0 beta.**
    - La librería con el formato de los datos de Google es **Google Gdata versión 2.0.**
    - La librería usada para poder lanzar el explorador para poder autenticar es **BrowserLauncher2 versión 1.3.**
    - Para la sincronización con Outlook se ha elegido la librería JNI **Jacob versión 1.16.**
- El sistema operativo utilizado es **Windows 7 versión Professional.**
- El sistema de correo utilizado es **Microsoft Outlook 2010.**
- El otro sistema de correo es **Gmail.**

Capítulo 3

Presupuesto

3.1 Resumen

Vamos a abordar un proyecto con el que podremos vender nuestros activos como empresa, y a parte vamos a construir una herramienta que facilite las tareas que no están asignadas como tal para los trabajadores comunes de una empresa cualquiera que utilice un ordenador como herramienta básica de trabajo.

Nos disponemos a crear un gadget con el que podamos tener sincronizadas todas las tareas que tenga que realizar un trabajador en su ámbito de trabajo, o las tareas diarias que tengan que ver con su vida personal. Las listas de tareas que nos disponemos a sincronizar pertenecen a los calendarios de Google y de Microsoft Outlook. De esta manera el trabajador no perderá un tiempo vital en pasar a mano una a una sus tareas de un calendario a otro, y así nosotros como empresa tampoco perderemos dinero en lo que lo lleva a cabo.

Procedemos a realizar, en este apartado, el presupuesto necesario para llevar a cabo el proyecto. Para calcular los costes del proyecto habrá que tener en cuenta tanto el coste de material como el del trabajo de las personas que han participado en su desarrollo.

3.2 Coste de personal

Los costes de personal incluyen los honorarios de los Ingenieros Técnicos en Informática de Gestión encargados del desarrollo del proyecto. La duración de este proyecto ha sido de 1 mes y 18 días. Suponiendo 20 días laborables al mes se obtiene un total de 38 días laborables. Con una jornada laboral de ocho horas diarias, la realización del proyecto ha requerido 304 horas aproximadamente. En la tabla se puede ver el resumen de costes directos de personal. Por tanto, el coste total asciende a 2.645,46 €.

Función	Categoría	Dedicación (hombres mes)	Coste hombre mes	Coste (Euro)
Desarrollador 1	ITIG	1,08	1.224,75	1.322,73
Desarrollador 2	ITIG	1,08	1.224,75	1.322,73
			Total	2.645,46

Tabla 3. 1 Coste personal

3.3 Coste de material

Los materiales empleados durante la realización del proyecto han sido los siguientes:

- Dos ordenadores con sistema operativo Windows 7, valorado aproximadamente en 700 euros cada uno.
    - Placa Base: Intel® Asrock Z77 Pro4
    - Procresador: Intel® Core™ i5 Processor i5-2500K
    - Disco Duro: HGST 4TB HDD SATA
- Un paquete Microsoft Office 2010 con un valor de 114,37€ con clave para 3 ordenadores.
- Conexión a Internet valorada aproximadamente en 40 euros al mes.

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste imputable
Ordenadores	1.400,00	100	2	48	58,33
Microsoft Office	114,37	50	2	48	2,38
Internet	40,00	100	2	3	26,67
				Total	87,38

Tabla 3. 2 Coste de material

3.1 Coste total

El presupuesto total para la realización de este proyecto está constituido por los costes de material y de personal presentados anteriormente. Como se observa en la tabla, **el presupuesto total asciende a 3.279,84€.**

Presupuesto Costes Totales	**Presupuesto Costes Totales**
Personal	2.645,46
Amortización	87,38
Costes Indirectos	547
Total	3.279,84

Tabla 3. 3 Coste total

Capítulo 4

Conclusiones y Trabajo futuro

4.1 Conclusiones

En este proyecto se ha desarrollado un gadget con un único objetivo: simplificar las tareas de los trabajadores, o al menos una de ellas, la organización. Como vimos en la definición de gadget que dimos ya unas cuantas hojas atrás, el objetivo del gadget era cumplir con un único objetivo, para el cual había sido desarrollado. Y creo que lo hemos conseguido.

Al principio del trabajo hemos hablado largo y tendido sobre los Gadgets, sobre los Widgets, sobre internet, y todo lo que implica, bueno, más bien una parte, porque si fuese todo, seria muy largo este trabajo. Pero al igual que los Gadgets, este, considero debe ser un trabajo con un único objetivo: Comprender que es un gadget y la mejor forma de hacerlo, a parte de estudiar un poco sobre el, es construir uno. Para ello primero hice un estudio, del que solo he planteado un par de cuadros resumen, en el que evaluaba las tecnologías que existen hoy en día para construir gadget y Widgets, y en ese trabajo la plataforma ganadora es, sin duda alguna, la empresa Google. Ya no solo por su forma de trabajar, si no por que la documentación que aportan esta muy cuidada, para que cualquier persona que esté en este mundillo, pueda utilizar sus herramientas y crear cosas impensables.

Pero, ¿Por qué he escogido este proyecto? Pues bien, la respuesta es que engloba un montón de tecnologías relacionadas con internet, así como poder crear un programa funcional para Windows acabado completamente. Gracias a este proyecto he podido terminar de aprender Java, de forma que me he tenido que enfrentar yo solo al problema y así poder dar solución. También he aprendido a buscar documentación de una empresa e indagar en las herramientas que te dan para poder solucionar el problema, como por ejemplo las librerías de Google. He tenido que tratar con Windows, de forma que he tenido que interactuar con uno de sus programas con Java, que se consigue con JNI, por lo que he podido establecer un puente entre Java y .NET. Y al fin realizar un programa con un instalador y que cree un icono visible tanto en el escritorio como en la barra de tareas al minimizarlo.

Creo que con este trabajo ya no se puede dudar de qué es un gadget y de todo lo que podemos explotar este tema, ya no solo en cuanto a las facilidades que podemos brindar a través de gadgets, si no de todo lo que nos pueden aportar, como por ejemplo, el mero hecho de emplazar un mini programa en el ordenador de cualquier persona, y que este mismo se convierta en un simple anuncio de nuestra empresa, como demostramos en otra parte de este trabajo.

En resumen, los gadgets se están convirtiendo en algo muy útil, y sobre todo en algo que según la naturaleza de la empresa que lo quiera desarrollar, se debe de tener en cuenta. Y más viendo como se desarrolla día a día el uso del móvil en nuestra vida cotidiana.

4.2 Trabajo futuro

Como posible trabajo futuro, una posibilidad es que nos olvidemos de Outlook y nos centremos en los demás sistemas operativos y sus gestores de correo predeterminados.

Consistiría en analizar los diferentes gestores de correo predeterminados en cada sistema operativo y realizar el mismo gadget para que funcione sin problemas en cada sistema operativo, de esta manera podríamos utilizar lo ya aprendido, y, en cierta manera, indagar un poco más para cada sistema operativo, y así poder generalizar este gadget para el resto de sistemas operativos del mercado.

Otra posibilidad podría ser un estudio para ver cuales son los gestores de correos utilizados por los usuarios de la web, y así poder proporcionarles este gadget en caso de que estos gestores no incluyan esta opción.

Anexo I

Manual de Instalación

La finalidad del presente capítulo es que cualquier usuario pueda instalar el proyecto. Para ello, en la primera sección se explicará paso a paso como configurar el entorno de desarrollo y en la segunda sección se explicará como ejecutar los proyectos para poder simular la aplicación.

AI.1 Configuración de la instalación

- **Java**

La versión de Java 6 Update 31 se puede descargar de http://java.com/es/download/windows_xpi.jsp?locale=es y elegimos donde queremos ubicar la descarga dentro de nuestro ordenador. Ejecutamos el instalador y aceptamos las condiciones de uso, y más tarde elegimos las partes que queremos que se instalen. En nuestro caso lo dejamos como esta y le damos a siguiente.

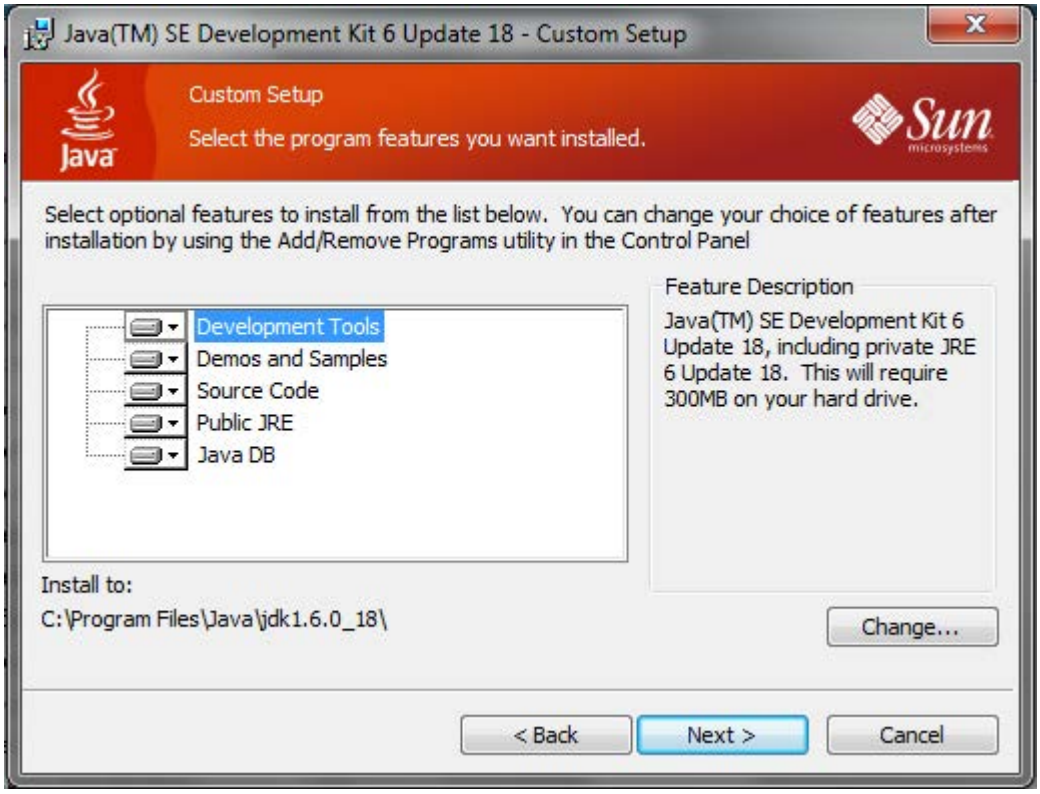


Figura AI. 1 Opciones Java

Una vez hecho esto te dará la opción de elegir donde ubicar la instalación de Java, elegimos y damos a siguiente. Se descargará a continuación el programa, y más tarde se instalará.

Figura AI. 2 Instalación Java

Para poder ejecutar este proyecto necesitamos **Eclipse Indigo versión 3.7.1**

- **Eclipse**

La versión 3.7.1 de Eclipse se puede descargar de http://www.eclipse.org/downloads/packages/eclipse-classic-372/indigosr2 y elegimos donde queremos ubicar la descarga dentro de nuestro ordenador.

Una vez descargado descomprimimos en una carpeta donde queremos que esté ubicado el eclipse para futuros usos. Para ejecutarlo solo hay que arrancar el fichero Eclipse.exe. Una vez arrancado lo único que nos pedirá es que le demos la ruta por defecto donde queramos que eclipse nos vaya guardando los proyectos que creemos:

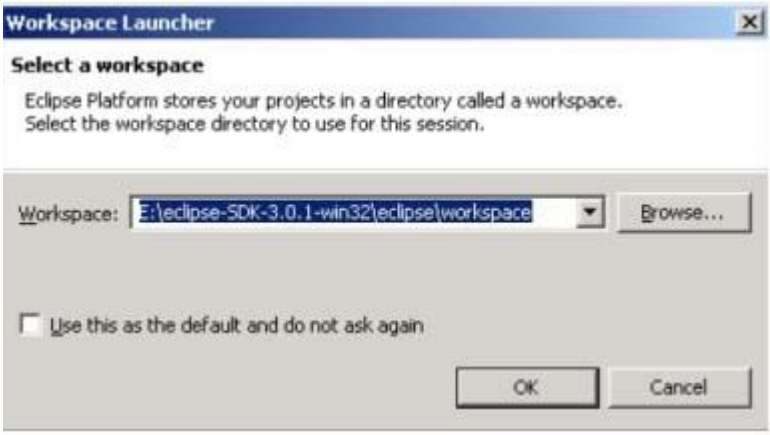


Figura AI. 3 Configuración workspace

Una vez instalado el Eclipse necesitamos instalar **Java Versión 6 Update 31** para que funcione este programa.

Una vez completado esto, ya podremos utilizar sin problemas el Eclipse. Así que ejecutamos el eclipse y procedemos a crear el proyecto y cargar las librerías necesarias.

Pulsamos File -> New -> Java Project

Escribimos el nombre del proyecto, elegimos en entorno JavaSE – 1.6 y elegimos la opción de crear carpetas separadas para recursos y clases.

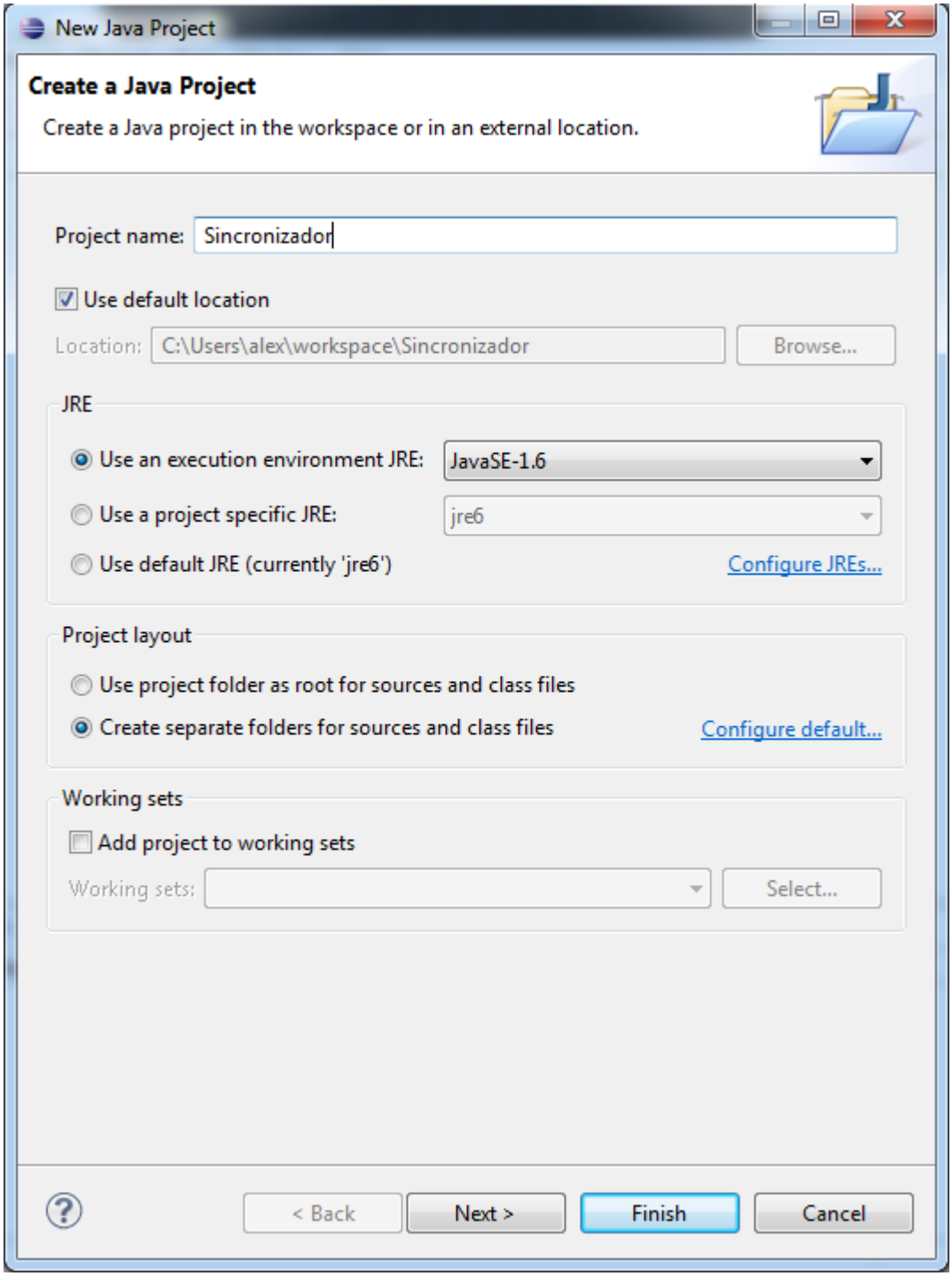


Figura AI. 4 Creación proyecto

Y listo, ya tenemos creado nuestro proyecto, ahora procedemos a cargar las librerías.

❖ Jacob

La versión de Jacob versión 1.16 se puede descargar de http://sourceforge.net/projects/jacob-project/ y elegimos donde queremos guardarla.

Una vez descargada lo descomprimimos en una carpeta y lo incluimos dentro del proyecto.

Seleccionamos el proyecto y pulsamos botón derecho y seleccionamos Propierties -> Java Build Path -> Add External JARs… y buscamos la carpeta donde tenemos la librería Jacob, y cargamos Jacob.jar

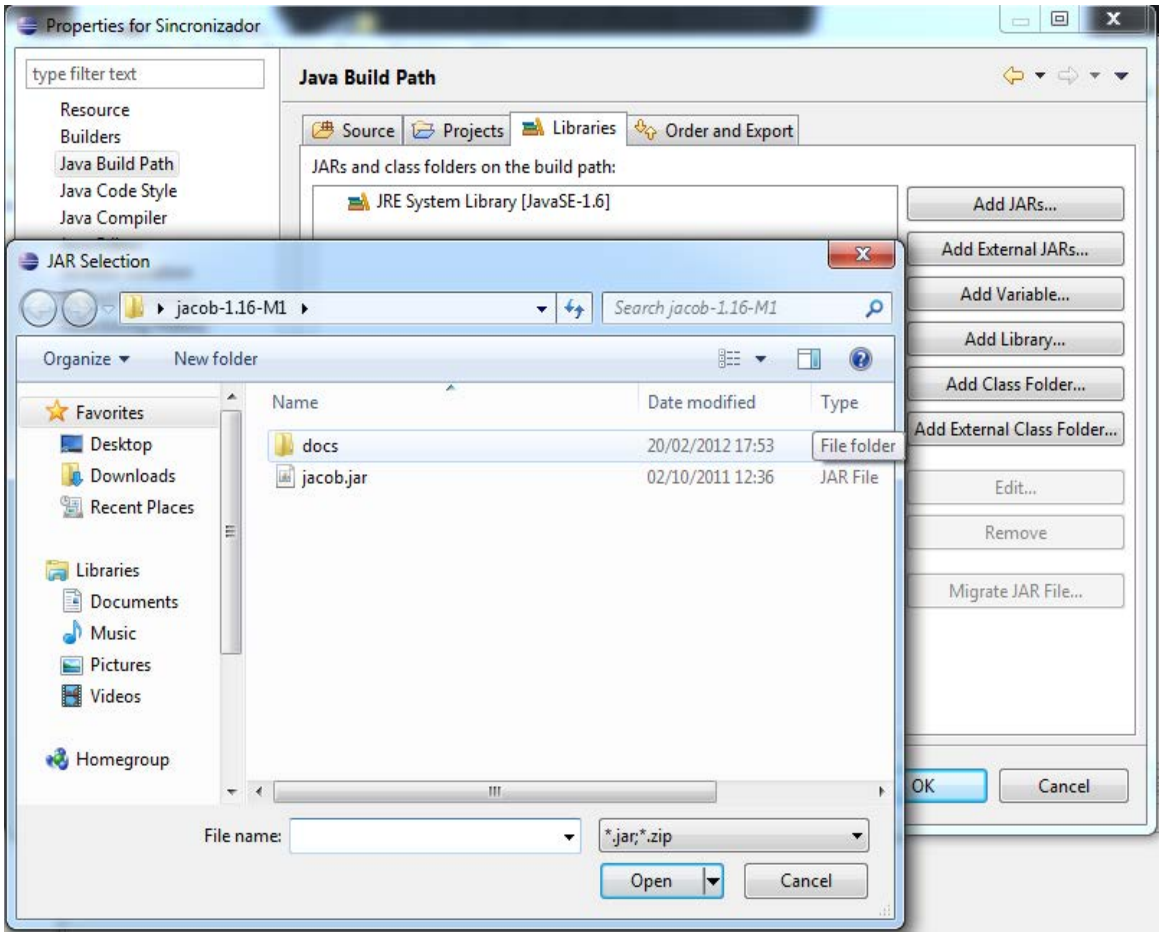


Figura AI. 5 Importar librería Jacob

Una vez cargado, tenemos que cargar la librería dll de Jacob para que pueda leer y escribir las tareas en el Outlook. Pulsamos en Jacob.jar y abrimos el desplegable, y en la sección Native library Location cargamos la carpeta donde está el dll, que es la carpeta de Jacob, y el mismo selecciona la librería dll necesaria según nuestro ordenador.

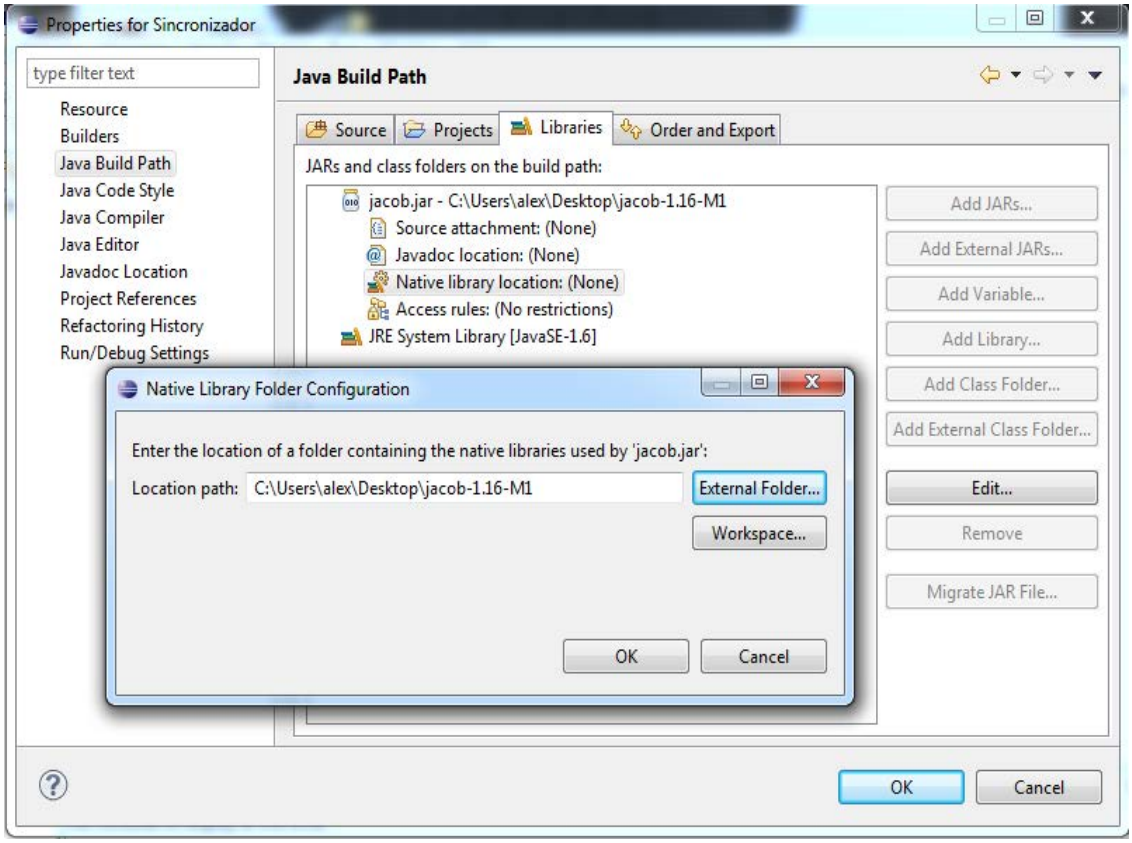


Figura AI. 6 Importar librería dll de Jacob

- **Resto de librerías.**

  - Descargamos la librería Gdata-Java-Client versión 1.46.0 de http://code.google.com/p/gdata-java-client/downloads/list y la incluimos dentro de Eclipse.

  - Descargamos la librería BrowserLauncher 2 de http://browserlaunch2.sourceforge.net/ y la incluimos en Eclipse.

  - Descargamos la librería Google-api-services-task versión 1.2.2 beta de http://www.java2s.com/Code/Jar/g/Downloadgoogleapiservicestasksv1122betajar.htm y la incluimos en Eclipse.

  - Descargamos la librería looks versión 2.2.0 de http://grepcode.com/snapshot/repo1.maven.org/maven2/com.jgoodies/looks/2.2.0 y la incluimos en Eclipse.

Tiene que quedar algo similar a esto:

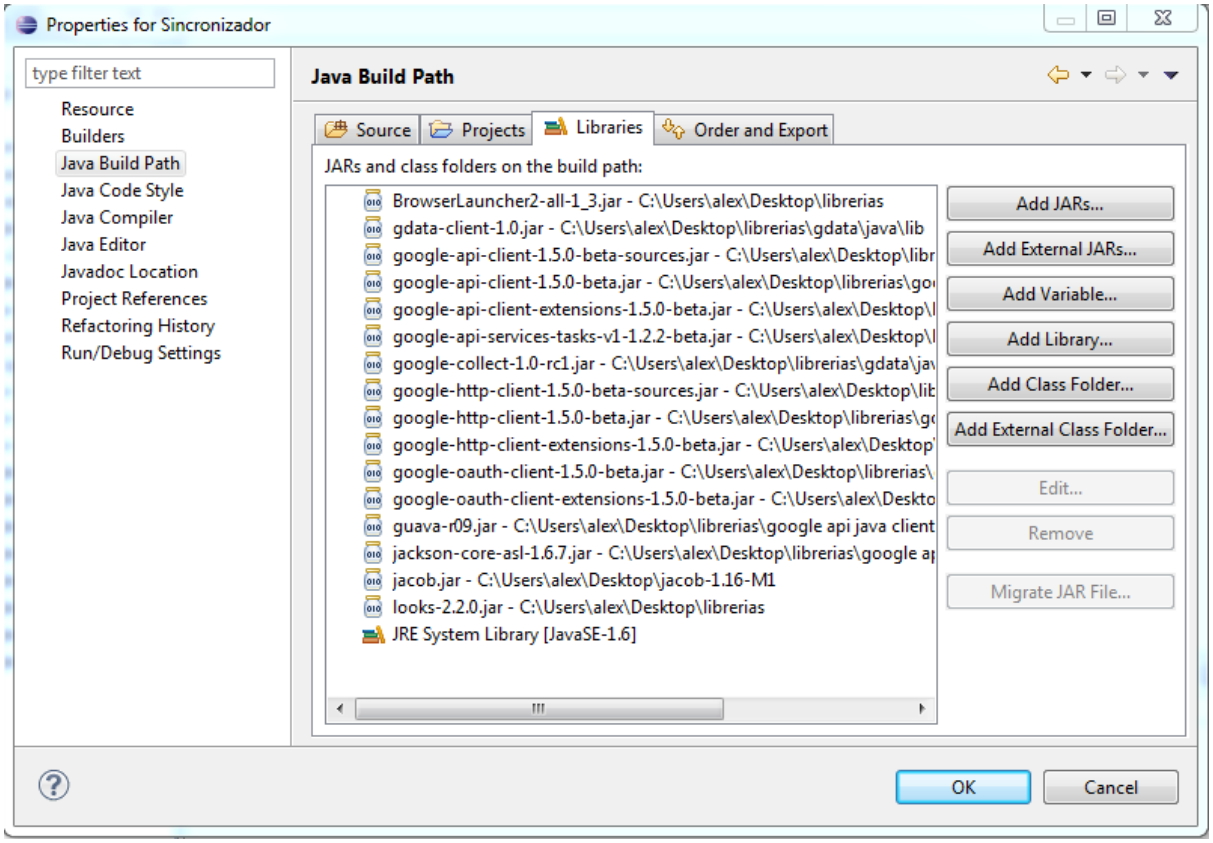


Figura AI. 7 Todas las librerías cargadas

- **Jigloo.**

Por ultimo descargamos Jigloo para poder crear la GUI de http://www.cloudgarden.com/jigloo/, en la pagina buscamos la dirección para poder descargarnos el programa desde Eclipse, y la dirección es esta **http://cloudgarden1.com/update-site.** Dentro del Eclipse seleccionamos Help -> Software Updates -> Find y en el cuadro de dialogo que pne Work with: introducimos la anterior dirección y se nos carga directamente Jigloo

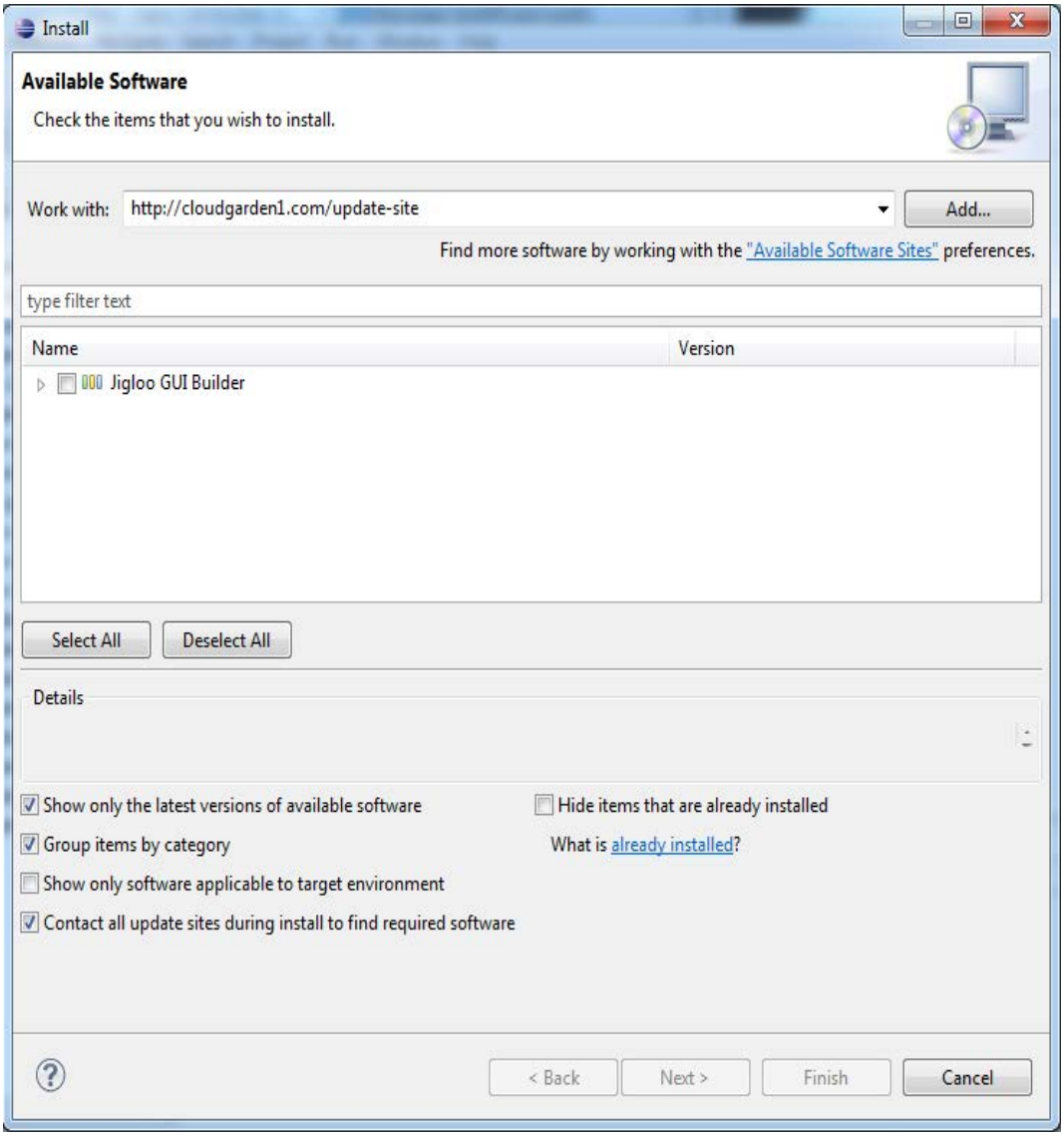


Figura AI. 8 Instalación Jigloo

Seleccionamos Jigloo y seguimos los pasos, hasta que quede instalado.

Ya esta tenemos todo lo necesario para poder realizar el proyecto deseado.

Anexo II

Manual de Usuario

En el apartado anterior, hemos visto como configurar todas las herramientas necesarias para poder crear nuestro programa, y en este apartado vamos a enseñar como funciona el programa para que no quede ninguna duda sobre su uso.

Una vez hemos instalado el gadget en nuestro ordenador el siguiente paso lógico es ejecutarlo. Y la pantalla principal es esta.

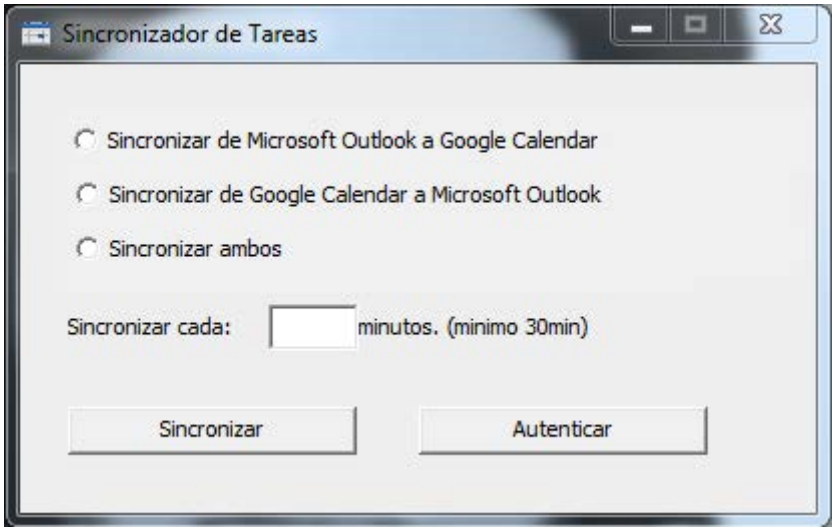


Figura AII. 1 Pantalla principal gadget

Como podemos comprobar tiene 3 opciones de sincronización disponible. Para elaborar este manual de usuario vamos a realizar la sincronización en ambos sentidos que es la opción más completa y a continuación establecemos el tiempo de sincronización cada 30 minutos. Este es el menor tiempo posible para evitar errores, que no tienen nada que ver con nuestro programa, sino por que Google, como restricción, no permite saturar al servidor con llamadas

constantes a los datos que contiene. Hemos incluido la opción de sincronizar al instante, y aconsejamos que no se abuse de ella para no entrar en este problema.

Si es la primera vez que ejecutamos la aplicación deberemos autenticarnos para que no tengamos problemas y podamos coger los datos que están contenidos en Google. Presionamos Autenticar y directamente el gadget nos lanza nuestro navegador de internet que tengamos configurado como predeterminado y carga directamente la pantalla para poder iniciar sesión en nuestra cuenta de correo.

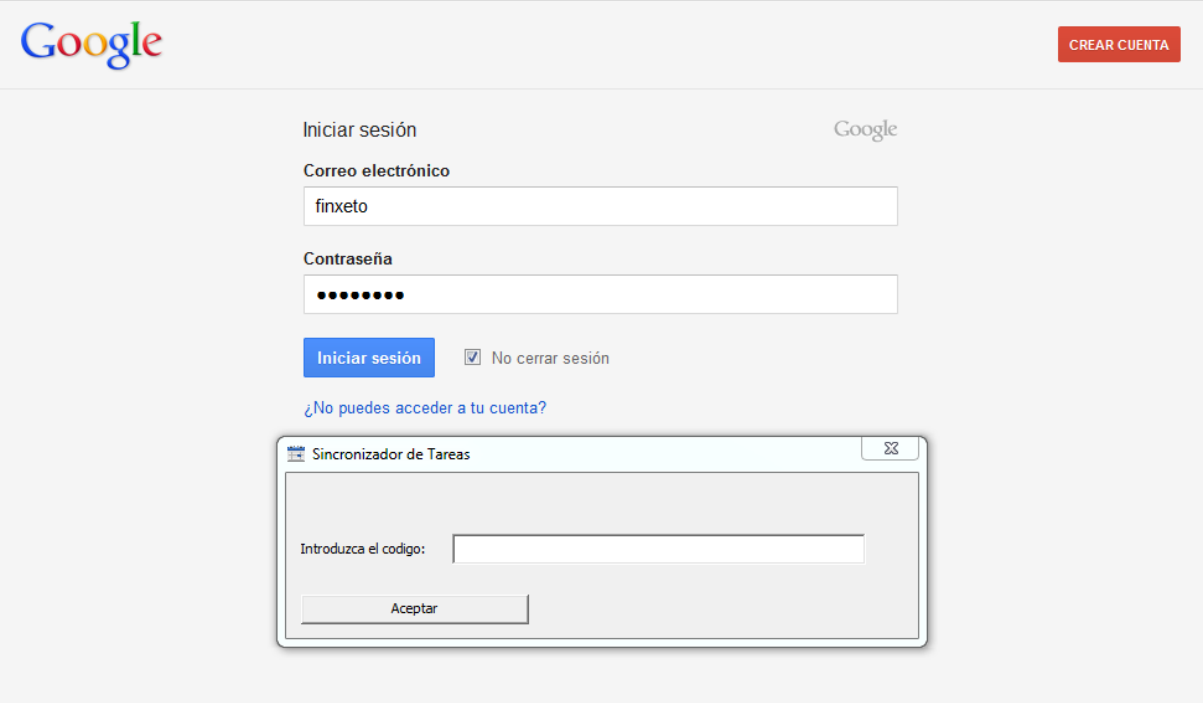


Figura AII. 2 Inicio sesión desde gadget

El gadget queda a la espera, con una nueva interfaz, para que podamos introducir el token, explicado anteriormente en el proyecto, para poder coger los datos necesarios relacionados con las tareas.

Al iniciar sesión nos preguntará si queremos permitir el acceso a estos datos, y seleccionamos el botón de Permitir acceso

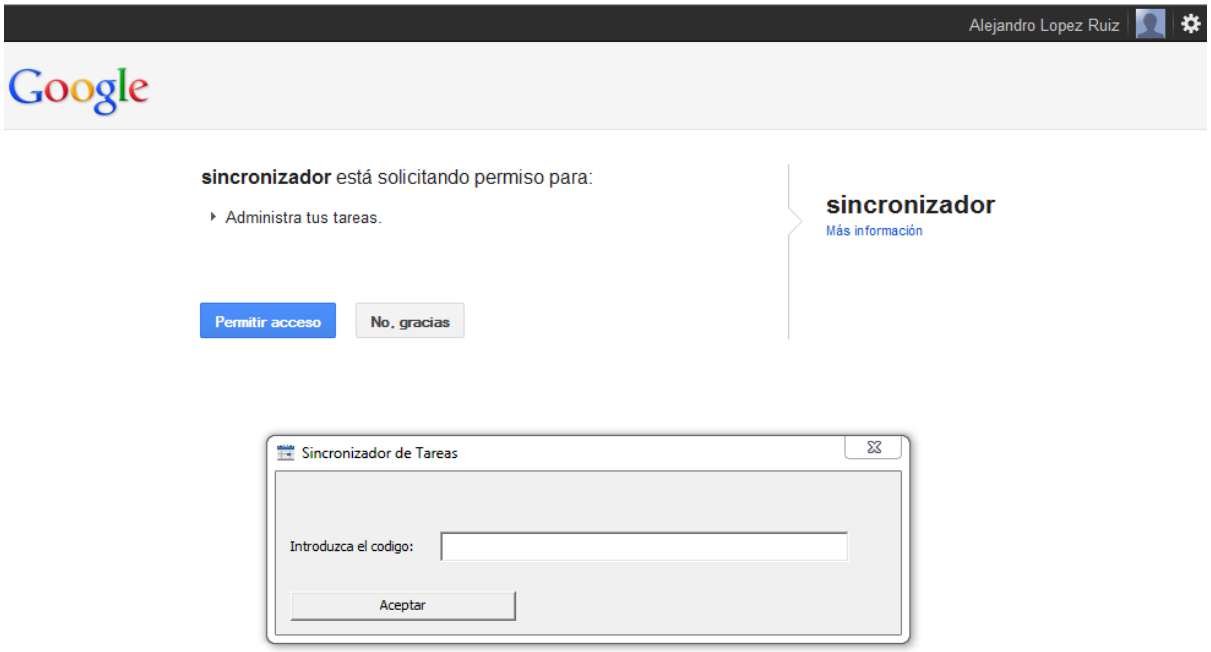


Figura AII. 3 Permitir acceso desde gadget

Al aceptar, al fin, nos mostrara el código que debemos introducir en el gadget.

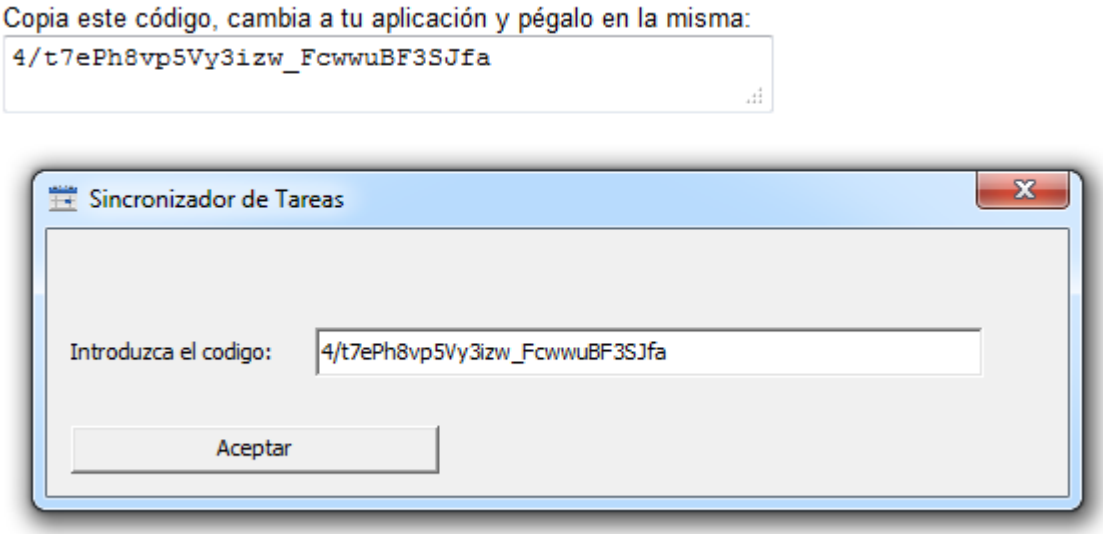


Figura AII. 4 Introducir código autenticación desde gadget

Y presionamos Aceptar. Vuelve el control a la interfaz principal para que seleccionemos las opciones comentadas anteriormente y ya estamos listos para poder sincronizar.

Ahora como lo que vamos a hacer es sincronizar, deberemos tener tareas que sincronizar, así que vamos a crearlas en ambos calendarios para ver si realmente funciona. Abrimos el Outlook y creamos 3 tareas:

- **Tarea 1:** Con la descripción “Soy la tarea 1”, con fecha a día de “07/03/2012” y que este completada.
- **Tarea 2:** Con la descripción “Soy la tarea 2”, con fecha a día de “07/03/2012” y sin completar.
- **Tarea 3:** Sin descripción, sin fecha y sin realizar.

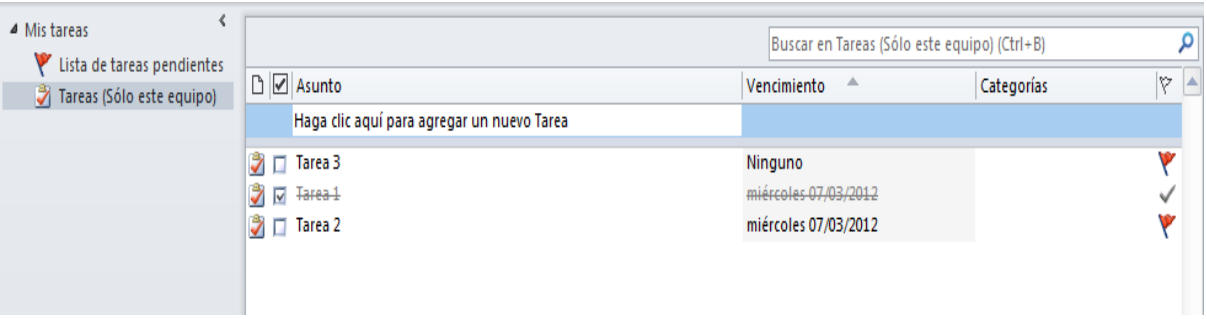


Figura AII. 5 Tareas Outlook

Creamos otras 3 tareas en el calendario de Google.

- **Tarea 1:** Con la descripción “Soy la tarea 1”, con fecha a día de “08/03/2012” y que este sin completar.
- **Tarea 2:** Con la descripción “Soy la tarea 2”, con fecha a día de “08/03/2012” y sin completar.
- **Tarea 4:** Sin descripción, sin fecha y sin realizar.

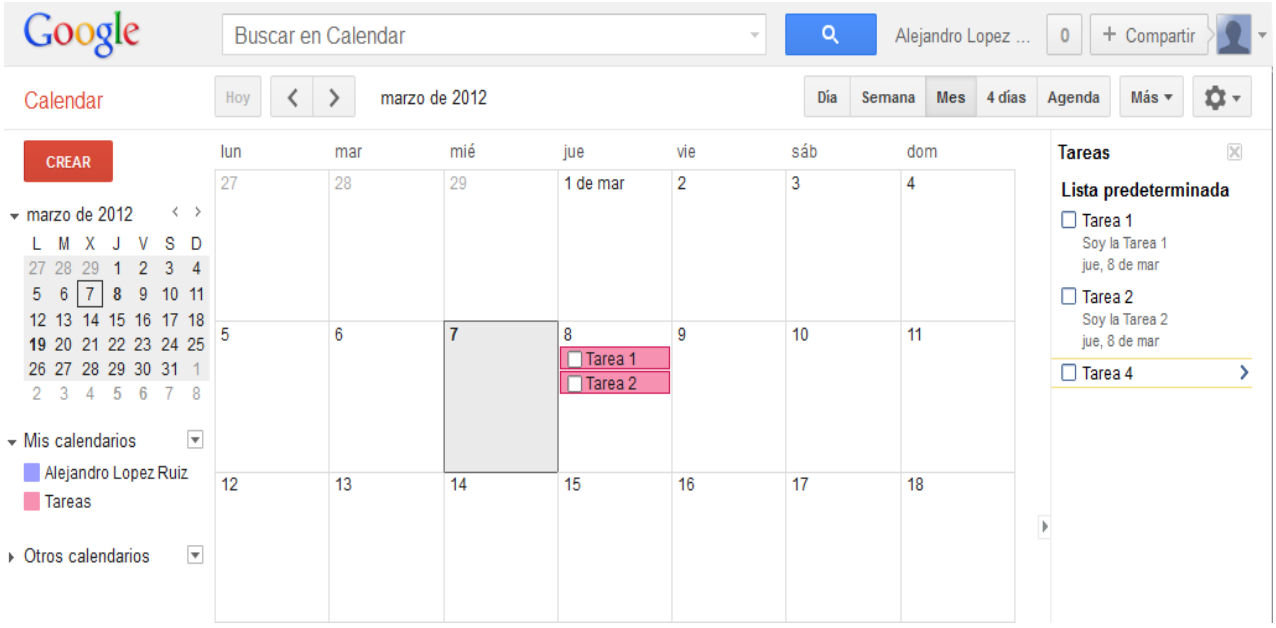


Figura AII. 6 Tareas Google

Y a continuación procedemos a sincronizar para ver los resultados y comentar como ha quedado.

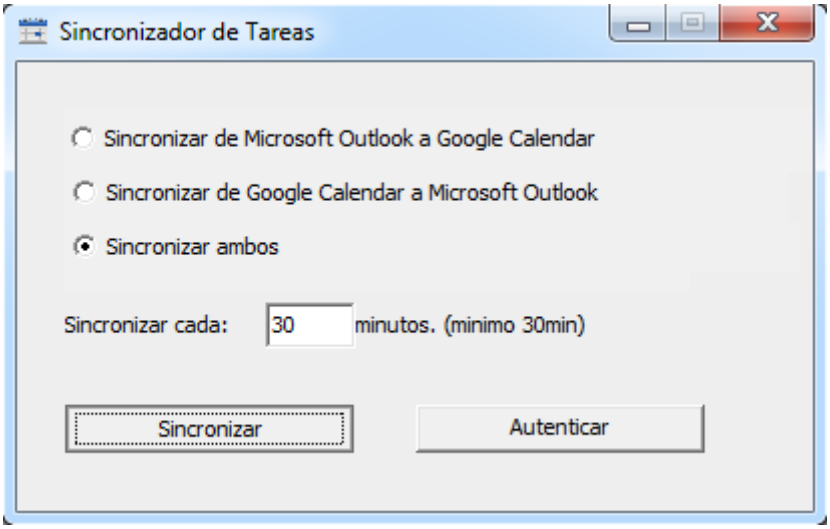


Figura AII. 7 Pantalla principal configurada

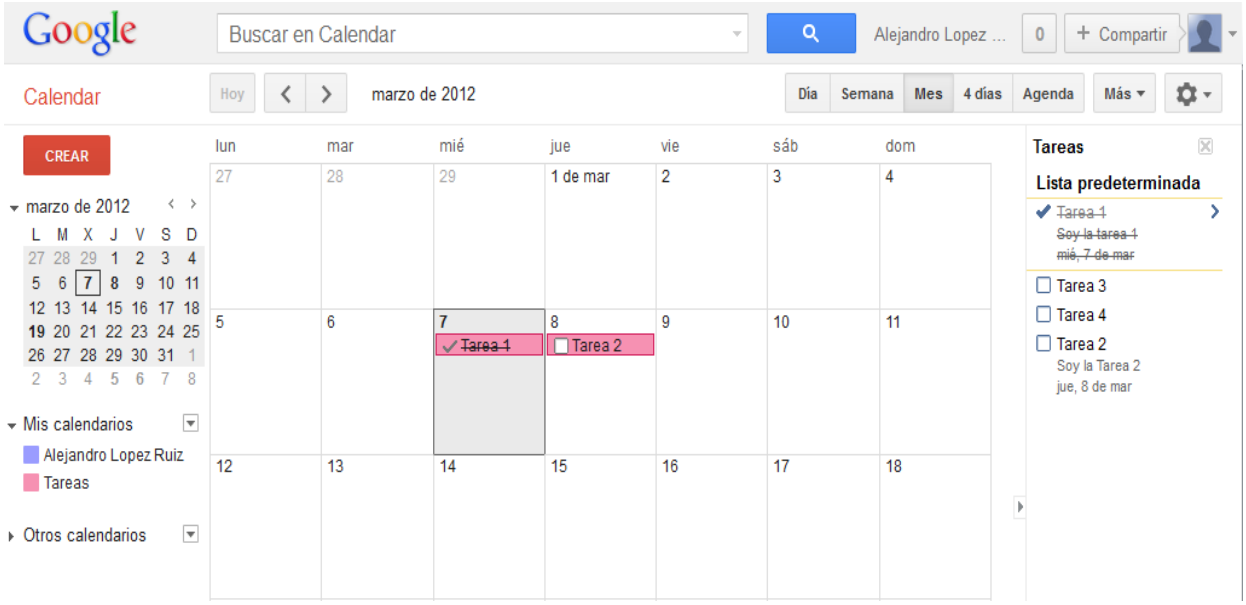


Figura AII. 8 Sincronización realizada en Google

Figura AII. 9 Sincronización realizada en Outlook

AII.1 Resultados

Como podemos comprobar han quedado exactamente igual y vamos a explicar que ha ocurrido para que estén iguales.

- **Tarea 1:** Por un lado en Outlook hemos creado la tarea con fecha "07/03/2012" y estaba completada, mientras que en Google hemos creado la misma tarea con fecha "08/03/2012" y sin completar. La prevalencia de tareas en este caso es que se copiará la tarea que está completa, sin importar la fecha que tenga. Por lo tanto es la que queda en ambos calendarios.

- **Tarea 2:** Hemos creado la misma tarea en ambos lados, pero en Outlook con fecha “07/03/2012” y en Outlook con fecha “08/03/2012”. Por lo tanto como ninguna de las dos esta completa, prevalece la tarea con la fecha más tardía.

- **Tarea 3 y 4:** En este caso se muestra como tareas que no están en ambos lados, sino que la tarea 3 está en Outlook y la tarea 4 está en Google, al final quedan ambas en los dos calendarios.

Si minimizamos el programa quedara escondido en la barra de sistema de Windows, y si presionamos el botón derecho de nuestro ratón sobre el icono del programa saldrá un pequeño menú con tres opciones:

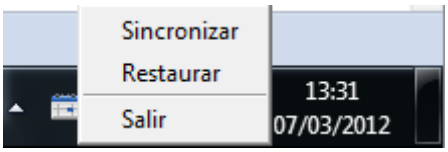


Figura AII. 10 Barra de sistema

La opción de Sincronizar, es la que sincroniza en el mismo instante, si necesidad de tiempo, pero como hemos comentado antes, hay que tener cuidado para no desembocar errores no deseados.

Anexo III

Orden de Prioridades en las tareas

Como ultimo anexo vamos a establecer las prioridades que tienen las tareas para que se sincronicen unas y no otras.

- Si dos tareas tienen la misma, o distinta fecha, y una esta completada y la otra no, siempre se sincronizará la tarea completada, mientras que la otra se borrara.

- Si dos tareas son iguales, en cuanto a nombre, con distintas fechas y están duplicadas en el mismo o en el otro calendario, se eliminará todas las tareas con fecha menor, y solo quedará una tarea con la fecha mayor. Se eliminan duplicidades.

- Si, por ejemplo, sincronizamos de Outlook a Google, y existen en ambos calendarios las mismas tareas, previamente sincronizadas pero creada inicialmente en Google, si eliminamos la tarea en Outlook, automáticamente se elimina la tarea en Google.

- Si, por ejemplo, sincronizamos de Outlook a Google, y existe una tarea creada recientemente en Google, esta tarea al terminar de sincronizar, seguirá estando en el mismo calendario a pesar de haber sincronizado las tareas de Outlook.

- Si tenemos sincronizada una tarea ya existente en ambos calendarios, y por ejemplo, sincronizamos de Outlook a Google, si la tarea es borrada de Google, y sincronizamos la tarea volverá a estar en Google ya que hemos sincronizado desde Outlook y de ahí no la hemos borrado. Si sincronizamos en sentido contrario, se eliminará de Outlook.

- La descripción de una tarea es inherente a la tarea, es decir, si una tarea prevalece a otra, no se salvará ni se mezclaran las descripciones.

Glosario

ARPA	Advanced Research Projects Agency
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
RIA	Rich Internet Applications
RSS	Really Simple Syndication
RDF	Resource Description Framework
XML	Extensible Markup Language
XHTML	Extensible HyperText Markup Language
API	Application Programming Interface
W3C	World Wide Web Consortium
CPU	Central Processing Unit
HTTPS	Hypertext Transfer Protocol Secure
URL	Universal Resource Locator
JSON	Notación de Objetos de JavaScript
JNI	Java Native Interface
SO	Sistema Operativo
JSDK	Java Standar Developer Kit
JRI	Java Runtime Interface
TCP/IP	Protocolo de control de transmisión/Protocolo de Internet
IPC	Comunicación entre Procesos
JDBC	Java Database Connectivity
Java IDL	Java Interface Description Language
JACOB	Java Com Bridge
COM	Component Object Model
DLL	Dynamic-link library

JVM	Java Virtual Machine
VM	Máquina Virtual
SQL	Structured Query Language
AWT	Abstract Window Toolkit
JFC	Java Foundation Classes
L&F	Looks & Feels
GUI	Interfaz Gráfica de Usuario
SWT	Standard Widget Toolkit
ITIG	Ingeniería Técnica en Informática de Gestión

Referencias

[1] Origen y evolución de Internet

http://www.nodo50.org/manuales/internet/1.htm Septiembre 2011

[2] Web 1.0

http://culturacion.com/2009/04/historia-web-10-pasado-y-realidad/ Septiembre 2011

[3] Web 2.0

Octavio Isaac Rojas Orduña.: "Web 2.0 manual no oficial de uso" (ESIC Editorial, 2007.)

Cristóbal Cobo Romaní y Hugo Pardo Kuklinski.: "Planeta web 2.0: inteligencia colectiva o medios fast food" (LMI, 2008.)

[4] Crowdsourcing

http://www.internauta.org.ar/index.php Septiembre 2011

[5] Microbloggings

Haizheng Zhang, Myra Spiliopoulou, Bamshad Mobasher, C. Lee Giles, Andrew McCallum, Olfa Nasraoui.: "Advances in Web Mining and Web Usage Analysis" (Springer, 2009.)

[6] Web Semántica

http://www.w3c.es/divulgacion/guiasbreves/websemantica Septiembre 2011

[7] Gadgets, Widgets, ¿Por qué construir un gadget? y plataformas

Sterling Udell.: "Pro Web Gadgets: Across Iphone, Android, Windows, MAC, Igoogle and More" (Apress, 2009.)

[8] Plataformas para novatos

http://www.masternewmedia.org/es/2010/03/04/como_crear_un_widget_guia_a_los.htm Septiembre 2011

[9] Yahoo Widgets

http://widgets.yahoo.com/ Septiembre 2011

[10] Google Gadgets

http://www.google.com/webmasters/gadgets/ Septiembre 2011

[11] Adobe Air

http://www.adobe.com/es/products/air/ Septiembre 2011

[12] Google web toolkit

http://code.google.com/intl/es-ES/webtoolkit/ Septiembre 2011

[13] Windows Sidebar

http://windows.microsoft.com/en-US/windows7/products/features/gadgets Septiembre 2011

[14] Mac Dashboard Widget

http://www.apple.com/downloads/dashboard/ Septiembre 2011

[15] Java

http://www.clubdesarrolladores.com/articulos/mostrar/38-java-su-historia-ediciones-versiones-y-caracteristicas-como-plataforma-y-lenguaje-de-programacion Febrero 2012

[16] Google APIs ClientLibrary para Java

http://code.google.com/p/google-api-java-client/ Febrero 2012

[17] OAuth 2.0

http://oauth.net/2/http://oauth.net/2/ Febrero 2012

[18] JSON

http://www.json.org/ Febrero 2012

[19] JNI

http://java.sun.com/docs/books/jni/html/titlepage.html Marzo 2012

[20] JACOB

http://danadler.com/jacob/

http://sourceforge.net/apps/mediawiki/jacob-project/index.php?title=Main_Page Marzo 2012

[21] Swing

es.scribd.com/doc/7222069/**Java-Swing** Marzo 2012

[22] Jigloo

http://www.cloudgarden.com/jigloo/ Marzo 2012